UNIVERSIDADE NOVA DE LISBOA

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica

Eficiência Energética

Sistema de Comunicação para Monitorar Consumos de Energia

Por

André Filipe Pereira Jorge

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientadores: Professor João Francisco Martins

Professor Luís Gomes

Lisboa

2010

Agradecimentos

Os meus sinceros agradecimentos, aos meus orientadores Prof. João Francisco Martins e Prof. Luís Filipe dos Santos Gomes pela credibilidade, apoio e disponibilidade, bem como todos os conselhos, sugestões e críticas deveras úteis para a realização deste trabalho.

Ao Mestre Pedro Miguel Ribeiro Pereira, pelo acompanhamento, disponibilidade, capacidade de trabalho e amizade. Muito obrigado.

Ao meu grande amigo André Bento pela amizade, incentivo e por tudo aquilo que passámos e passaremos juntos, um grande abraço.

Dedico este trabalho aos meus pais por tudo o que fizeram por mim, em particular pelo esforço financeiro que fizeram ao longo dos últimos anos, sem o qual não seria possível chegar aqui. A vocês, o meu amor, carinho e o maior obrigado que possa existir.

Aos meus avós que felizmente fazem parte da minha vida, obrigado pelos sorrisos, amizade, carinho e incentivo. Obrigado pelo apoio.

Um especial obrigado ao meu "Manhas"- António Mesquita Pereira -, por tudo aquilo que representas para mim, pelo teu olhar, criatividade e maneira de ser. Muito do que sou devo-o a ti.

Finalmente, quero agradecer com todo o amor e carinho à Joana. Obrigado por estares sempre comigo em todos os momentos, fáceis e difíceis. Espero compensar-te devido à pouca disponibilidade que tenho tido.

A ti Sokkokum….

Índice

Lista de Figuras

Lista de tabelas

Lista de Acrónimos

PPP: Princípio do Poluidor Pagador

CAN: Controller Area Network

CIA: CAN in Automation

ISA: International Society for Mesurement and Control

PROFIBUS: Process Field Bus

PIS: Protocolos Industriais Standardizados

CLP: Conrolador Lógico Programável

M/E: Mestre/Escravo

ASCII: American Standard Code for Information Interchange

RTU: Remote Terminal Unit

CPU: Central Processing Unit

PIC: Programmable Interface Controller

Resumo

Com vista à preservação do meio ambiente, a Comunidade Europeia estabeleceu normas restritas relativamente ao consumo de energia. Para cumprir tais metas será útil que os consumidores - tanto de pequenos como de grandes complexos – possam avaliar os seus diagramas de consumo de forma a planearem ou reajustarem os seus hábitos energéticos. Isto leva à necessidade de um sistema que permita monitorizar a energia de vários consumidores descentralizados e agregar todas as informações recolhidas num único sítio, evitando assim a necessidade de efectuar leituras locais.

Neste documento é descrito o desenvolvimento de um sistema em código aberto para monitorização e aquisição de dados de diversos analisadores de energia. O sistema desenvolvido é baseado num conjunto distribuído de contadores de energia eléctrica e num computador com ligação Internet/Intranet através de RS485 usando o protocolo de comunicação Modbus RTU.

O sistema foi desenvolvido para grandes complexos de edifícios e foi validado no campus da Faculdade de Ciências e Tecnologia. Considera duas aplicações distintas. A primeira permite ao utilizador verificar, em tempo real, o consumo energético em qualquer ponto do complexo, bem como elaborar relatórios, produzir diagramas de carga e enviar emails com os dados. A segunda aplicação mantém os registos de consumo de energia activa/reactiva a fim de se verificar a existência de alguma situação anómala. Pode também ser usada para facturar os gastos de cada departamento em energia.

Abstract

Strict regulations, regarding power consumption and energy efficiency, have been set by the European Community to preserve the environment. To fulfil energy conservation goals, it will be helpful if consumers could assess their energy load diagram so that they can plan/re-plan their energy consumption profile. This leads to the need of a system to monitor the energy consumption both for small and large complexes.

In this thesis is described the development of an open source system for monitoring and data acquisition of several energy analyzers. The developed system is based on a computer with Internet/Intranet connection by means of RS485 using Modbus RTU as communication protocol.

The monitoring/metering system was developed for large building complexes and was validated in the Faculdade de Ciências e Tecnologia University campus. The system considers two distinct applications.

The first one allows the user to verify, in real time, the energy consumption of any department in the complex, produce load diagrams, tables and print, email or save all available data. The second application keeps records of active/reactive energy consumption in order to verify the existence of some anomalous situation, and also monthly charge each corresponding department.

Capítulo 1 - Introdução

Neste capítulo é feita uma introdução à Eficiência Energética, tanto em Portugal como na Europa. Aborda-se a problemática da má gestão da energia na indústria e apresentam-se algumas soluções. No final da secção são revelados os objectivos do trabalho e a estrutura da dissertação.

1.1 Motivação

1.1.1 Eficiência Energética em Portugal e na Europa

A economia portuguesa caracteriza-se por possuir uma elevada intensidade energética e carbónica, bem como pelo facto de ser fortemente dependente da importação de energia primária (cerca de 85% do total necessário, especialmente petróleo), Figura 1.1.

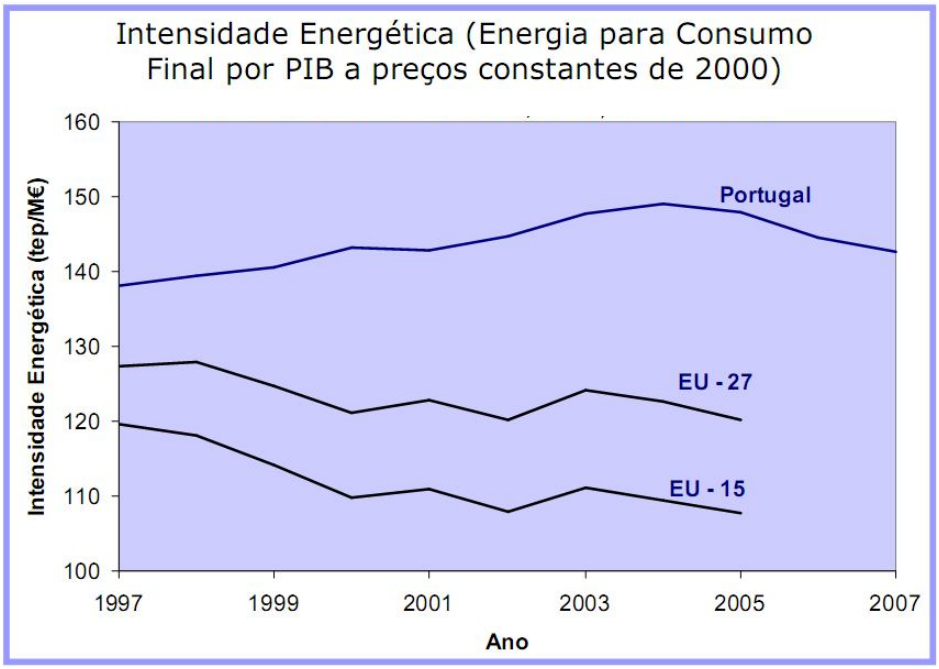


Figura 1.1 – Intensidade Energética [1]

A evolução da intensidade energética de Portugal divergiu significativamente da média europeia na última década. No entanto, existe uma inversão da tendência relativamente ao período 2005-2007, [1]. Esta inversão deve-se ao efeito cumulativo da diminuição do consumo de energia final e do aumento do PIB.

A emissão excessiva de dióxido de carbono e de outros gases com efeitos de estufa é uma das principais consequências da falta de eficiência no consumo de energia obtida da queima de combustíveis fósseis, Tabela 1.1.

Tabela 1.1 – Valor das emissões de CO_2 em Portugal e na EU-15 [2]

Valor das Emissões de CO_2 em Portugal e na EU-15

	Emissões de CO_2 (Mt)				% acima da Meta de Quioto (em 2005)
	2003	2004	2005	Meta Quioto 2012	
Portugal	83,7	84,6	85,5	76,2	**12,3**
UE-15	4215	4227	4192	3925	**6,8**

Entre 2003 e 2005, observou-se, em Portugal, um aumento das emissões de CO_2 em contraste com a ligeira diminuição observada para os países da União Europeia.

De acordo com a Tabela 1.1, os esforços que Portugal tem efectuado para reduzir as emissões de CO_2 têm tido pouco sucesso.

A continuação da divergência relativamente às metas de Quioto para 2012 acarretará, para além dos prejuízos ambientais, consideráveis prejuízos económicos para Portugal, [2].

Os sectores dos transportes e dos serviços são, segundo a Eurostat, os que mais contribuíram para o aumento do desvio da intensidade energética para a média europeia. Por outro lado, a indústria foi o único sector que contribuiu para a sua redução, Figura 1.2.

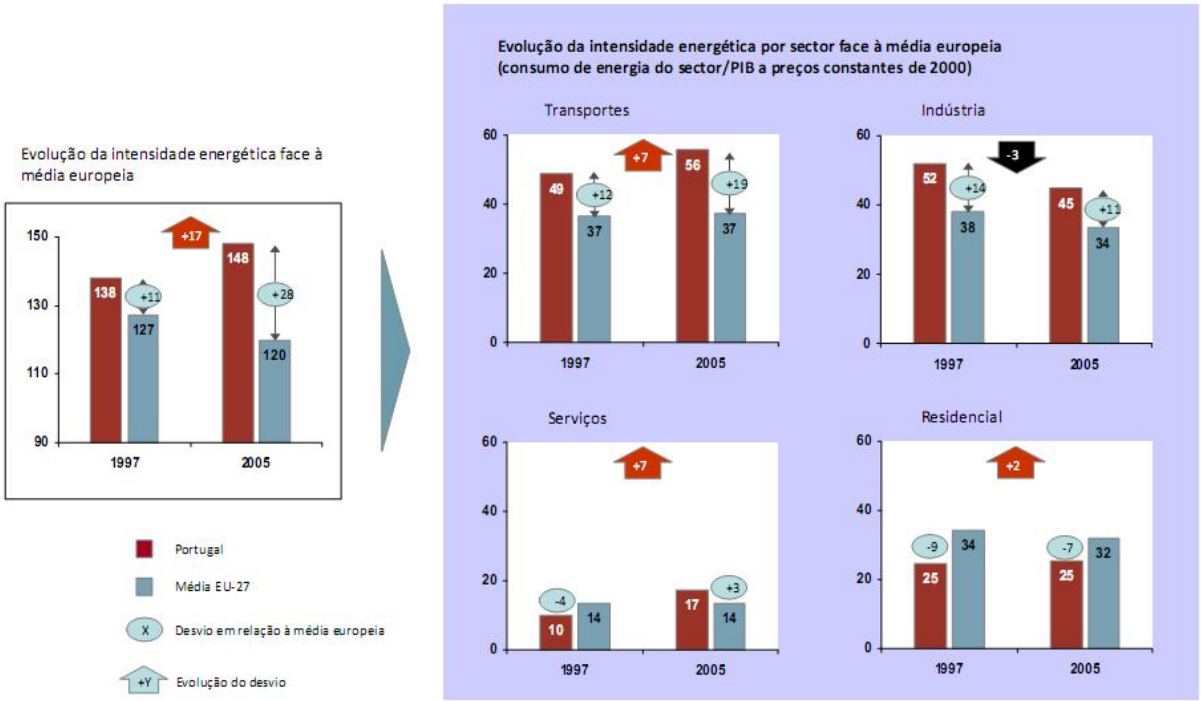


Figura 1.2 – Evolução da intensidade energética por sector face à média europeia [3]

Posto isto, é necessário efectuar esforços mais ambiciosos e dinâmicos na promoção da eficiência energética (EE) em todos os sectores da sociedade Portuguesa. Para tal, surge o PNAEE, Plano Nacional de Acção para a Eficiência Energética, [3].

O PNAEE tem como objectivos a implementação de medidas políticas que regulamentam o consumo energético e as emissões dos gases de efeito de estufa. Estas medidas políticas devem ainda fomentar/estimular a aplicação de medidas tecnológicas que poupam energia.

Relativamente a Portugal, a convergência com o nível de intensidade energética europeu necessita de ser acelerada através de um PNAEE, denominado Portugal Eficiência 2015. No âmbito deste plano foram definidos 12 Programas abrangentes para actuar nas várias vertentes da eficiência energética, Figura 1.3.

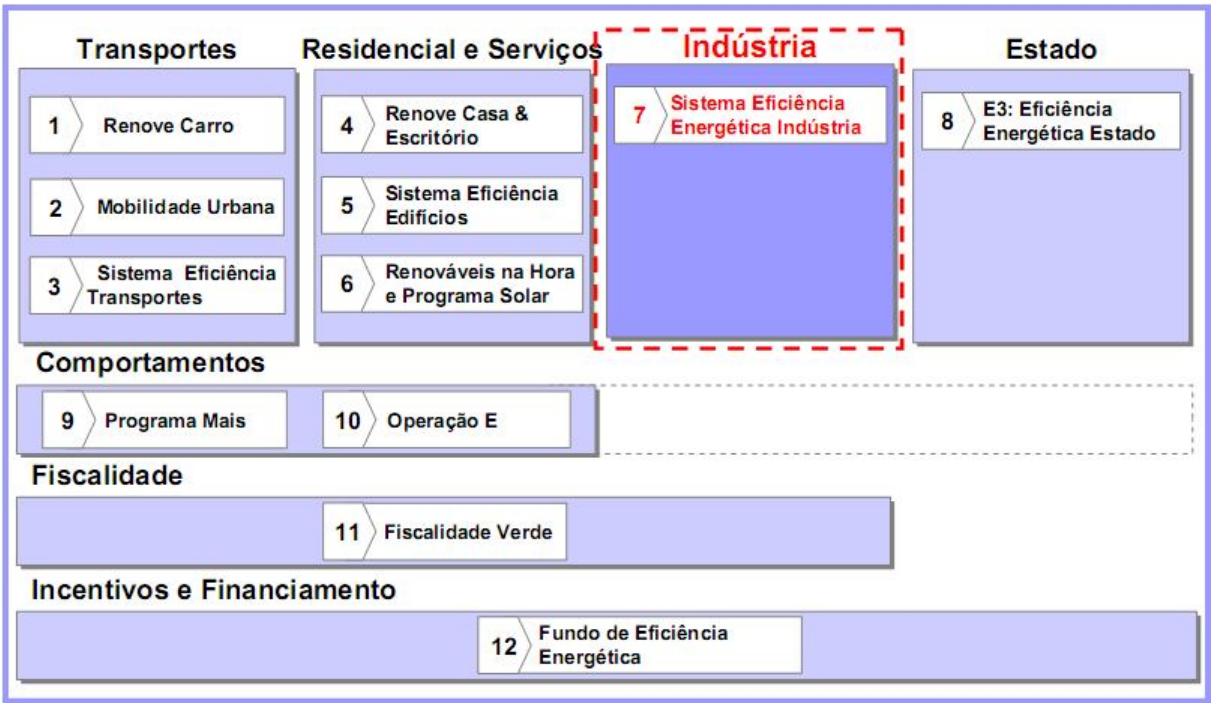


Figura 1.3 – Os 12 Programas do PNAEE Portugal Eficiência 2015 [3]

As medidas políticas e tecnológicas para aumentar a EE da indústria portuguesa (Programa 7) devem seguir as recomendações e directrizes gerais da União Europeia - ou não fosse Portugal um estado-membro.

O objectivo das medidas implementadas pelo PNAEE é um aumento de 10% na eficiência energética global (todos os sectores da sociedade) até 2015.

A operacionalização do plano implica a criação de um Fundo para a Eficiência Energética e um acompanhamento eficaz e articulado com o Plano Nacional para as Alterações Climáticas (PNAC).

O Programa 7 - Sistema de Eficiência Energética na Indústria – levou à criação de um Sistema de Gestão dos Consumos Intensivos de Energia (SGCIE) e de um Programa para a Energia Competitiva na Indústria.

O SGCIE engloba auditorias energéticas obrigatórias, com inclusão de um Plano de Racionalização do Consumo de Energia (PREn) e relatórios de execução e progresso bianuais. São estabelecidas metas relativas à intensidade energética e carbónica a atingir, no âmbito do PREn e obrigatoriedade de implementar medidas com *payback* mais curto. Em caso de não cumprimento das metas e medidas, as empresas deverão pagar uma penalidade ou reembolsar os apoios recebidos e os benefícios de isenção de ISP.

O Programa para a Energia Competitiva na Indústria apoiará os operadores com Acordos de Racionalização dos Consumos de Energia tais como: isenção do ISP, ressarcimento parcial dos custos com auditorias energéticas e ressarcimento de 25% dos investimentos em equipamentos e sistemas de gestão e monitorização de consumos de energia [3].

1.1.2 Eficiência Energética na Indústria

O peso da factura energética nos custos de exploração duma empresa do sector industrial é habitualmente baixo, quando comparado com o peso de outros factores de produção, nomeadamente mão-de-obra e matéria-prima. A gestão de energia é por isso frequentemente negligenciada, facto que gera significativos desperdícios de energia e contribui para a redução da competitividade das empresas.

Adicionalmente, continua presente na mente de alguns industriais a ideia de que o crescimento económico acarreta necessariamente um aumento dos consumos de energia [3]. O conceito de Utilização Racional de Energia (URE), surgido no seguimento dos chamados choques petrolíferos, veio alterar decisivamente a forma de encarar a energia, demonstrando ser possível crescer sem aumentar os consumos ou afectar a qualidade da produção. A chave da questão designa-se gestão de energia. Como qualquer outro factor de produção, a energia deve ser gerida contínua e eficazmente.

Embora o argumento da competitividade continue naturalmente a ser aquele que mais sensibiliza a generalidade dos industriais, a crescente pressão ambiental veio reforçar a necessidade de utilizar eficientemente a energia. Seja por imposição legal, seja pela necessidade de cumprir requisitos ambientais como forma de aceder a sistemas de apoio ou simplesmente por uma questão de imagem ou pressão da opinião pública, cada vez mais a eficiência energética está na ordem do dia. É para além disso unanimemente aceite que, mais cedo ou mais tarde, instrumentos políticos de mercado, como taxas ou impostos ambientais, introduzirão finalmente o Princípio do Poluidor Pagador (PPP), penalizando fortemente as empresas menos preparadas [3].

É assim que assumem particular importância o levantamento e a auditoria energética. Com efeito, qualquer processo de gestão de energia terá necessariamente que começar pelo conhecimento da situação energética da instalação. O princípio é óbvio - para gerir é indispensável conhecer o objecto de gestão.

O levantamento energético pode interpretar-se como a primeira radiografia ao desempenho energético da unidade fabril. Através dele, avalia-se quanta energia é efectivamente consumida e de que forma é essa energia utilizada, estabelecem-se os principais fluxos e identificam-se os sectores ou equipamentos onde é prioritário actuar.

Por auditoria energética entende-se o exame detalhado das condições de utilização de energia na instalação. A auditoria permite conhecer onde, quando e como a energia é utilizada, qual a eficiência dos equipamentos e onde se verificam desperdícios de energia, indicando igualmente soluções para as anomalias detectadas.

A auditoria energética pode também constituir uma obrigação legal. Com efeito, estão abrangidas pelo Regulamento de Gestão do Consumo de Energia (R.G.C.E.), todas as empresas ou instalações consumidoras intensivas de energia.

A auditoria energética surge assim como um instrumento fundamental, que o gestor de energia possui para contabilizar os consumos de energia, a eficiência energética dos seus equipamentos e as perdas que se verificam, tendo como finalidade última reduzir essas perdas sem afectar a produção, isto é, economizar energia através do uso mais eficiente da mesma.

A necessidade de determinar os consumos de energia sob diferentes formas, constitui um requisito básico para a realização de uma auditoria energética. Controlar os fluxos de energia que entram, circulam e saem da empresa é extremamente importante para quantificar as perdas de energia [4].

As diversas formas de energia adquiridas pela empresa auditada são conhecidas, uma vez que são medidas e facturadas pela empresa fornecedora. No entanto a desagregação dos consumos por utilização, secção ou equipamento e a avaliação das perdas de energia necessita ser contabilizada, ou seja, medida.

A maioria das empresas não dispõe ainda de instrumentação de medida adequada à realização das medições e registos necessários [5]. Para além disso, quando existe, a instrumentação instalada tem por vezes uma precisão desconhecida ou duvidosa. É frequente encontrarem-se em muitas empresas, determinados equipamentos ou sectores responsáveis por uma grande parte do consumo global, sem que tenham contadores instalados, o que impossibilita a determinação dos respectivos consumos específicos bem como a detecção de situações de consumos anómalos. Torna-se assim indispensável recorrer ao uso de instrumentos ou soluções de monitorização de energia que produzam resultados fiáveis para uma boa análise da instalação em causa.

Os sistemas de monitorização de consumos de energia permitirão realizar os cálculos dos consumos específicos de energia por equipamento, por produto, por sector produtivo e o global da instalação, revelando-se como um objecto fundamental para uma boa análise energética. São instrumentos que poderão servir de apoio ao gestor de energia, devido à informação por eles fornecida, permitindo tirar conclusões relativamente às grandezas eléctricas características de um circuito eléctrico de uma instalação ou equipamento, como por exemplo: o factor de potência, o diagrama de carga, o índice de carga dos transformadores, o equilíbrio entre fases, etc.

Com a implementação deste tipo de sistemas de gestão de energia, pode-se dizer que as economias típicas alcançadas e decorrentes exclusivamente do processo da constante monitorização e controlo dos consumos de energia reais e do consumo de energia padrão, são de 3% nos consumos eléctricos e 5% para as restantes formas de energia [5].

1.2 Objectivos

A eficiência energética não pode ser desassociada do termo "Utilização Racional da Energia" (URE), que consiste num conjunto de acções e medidas, que têm como objectivo a melhor utilização da energia, tanto no sector doméstico, como no sector de serviços e industrial.

A URE torna-se assim um factor importante de economia energética e redução de custos. Tendo em consideração uma série de recomendações e conselhos úteis, é possível reduzir os consumos energéticos mantendo o conforto e a produtividade das actividades dependentes de energia.

O trabalho proposto tem por objectivo desenvolver um sistema que permita fazer a monitorização de consumos de energia numa determinada instalação. O sistema a desenvolver deve permitir:

- Monitorização de consumos de energia;
- Gerar relatórios de consumo de energia e custos associados;
- Alarmes;
- Controlo de ponta;
- Permitir uma melhor gestão de energia;
- Permitir monitorar o consumo de energia de vários contadores a partir de um único local.

O trabalho em causa tem um fim específico: desenvolver um sistema de gestão de consumos para a FCT. A FCT tem instalado em vários pontos do Campus, algumas dezenas de contadores de energia. Estes contadores possuem uma interface de comunicação RS 485 que utiliza o protocolo Modbus.

Neste trabalho é pretendido que se desenvolva o sistema de comunicação com a aplicação anterior para uma utilização em rede, de modo a permitir a comunicação com todos os contadores instalados no Campus.

1.3 Estrutura da Dissertação

Além da Introdução, a dissertação é composta por quatro capítulos adicionais.

No Capítulo 2 - Monitorização de Consumos – é feita uma análise ao estado da arte, no que respeita a soluções de monitorização de consumos de energia presentes no mercado.

No capítulo da Comunicação, Capítulo 3, é feita uma breve introdução à Automação Industrial, passando depois pelos diversos protocolos de comunicação e acabando por enfatizar o protocolo escolhido para implementar este trabalho. No final da secção, descreve-se a interface criada e a forma como os dados de consumo energético são recolhidos e analisados.

A Implementação é o título do Capítulo 4. Aqui apresenta-se o hardware usado para implementar a rede, tal como os detalhes da implementação. É apresentado um exemplo de aplicação e os dados experimentais recolhidos, tal como a análise dos mesmos.

O Capítulo 5, Conclusões, conclui esta exposição. É feita uma síntese do trabalho e das contribuições inerentes. Finalmente, são referidas algumas perspectivas de trabalho futuro.

Capítulo 2 -
Monitorização de Consumos

Os sistemas de monitorização de consumos de energia existentes no mercado são apresentados neste capítulo. Em plano de destaque estarão os protocolos de comunicação por eles utilizados. Será ainda apresentada a metodologia de trabalho utilizada como linha de pensamento para o cumprimento dos objectivos já especificados.

2.1 Definição e Conceitos – Apresentação da Metodologia de Trabalho

Um bom ambiente de trabalho, produtivo para a instituição e agradável para os colaboradores, favorece não só o desempenho da empresa como também a sua evolução, tornando-a mais competitiva.

A chave para esta evolução baseia-se nas tecnologias de informação e no conforto ambiental, às quais só se tem acesso devido à energia eléctrica. Porém, há que ter em conta que o custo da energia é cada vez maior. Assim, para se manter na frente da competição, todos os custos de uma empresa devem ser minuciosamente contabilizados, sobretudo no que toca aos gastos energéticos [5].

Posto isto, a implementação de sistemas de monitorização de consumos de energia é cada vez mais importante. Estes sistemas fornecem aos gestores das empresas uma visão global e desagregada dos consumos e respectivos custos dispendidos.

Um sistema de monitorização fiável e eficiente levará à optimização dos custos de exploração da instalação e dos equipamentos. Poder-se-ão analisar de forma precisa os equipamentos mais importantes (muitas vezes são equipamentos bastante sensíveis e caros e que se podem avariar devido a picos ou falhas de tensão), identificar desperdícios de consumo, efectuar contabilizações energéticas e, desta forma, efectuar uma melhor gestão da instalação.

Outras vantagens associadas a este tipo de sistemas prendem-se com a escolha mais favorável dos tarifários disponíveis no mercado, alerta dos serviços de manutenção para consumos anómalos, imputação de custos e sensibilização ambiental.

Os sistemas de monitorização de consumos existentes no mercado, quando aplicados a grandes complexos, revelam-se bastante dispendiosos. O mesmo se pode dizer relativamente à monitorização tradicional – feita por um operário – pois existe a necessidade de destacar alguém para esse trabalho e, muitas vezes acarreta erros de medição e, não é tão precisa como a monitorização automática.

Posto isto, existe a necessidade de criar um sistema de baixo custo que, consiga recolher toda a informação relativamente a uma instalação eléctrica conduzindo à sua optimização e consequente redução de custos.

Inicialmente, houve necessidade de conhecer o estado da arte relativamente aos sistemas de monitorização de consumo de energia, quais as soluções existentes no mercado, as suas vantagens e desvantagens, custos inerentes e aplicabilidade a pequenos e grandes complexos. A Figura 2.1 mostra o ciclo de intervenção a seguir.

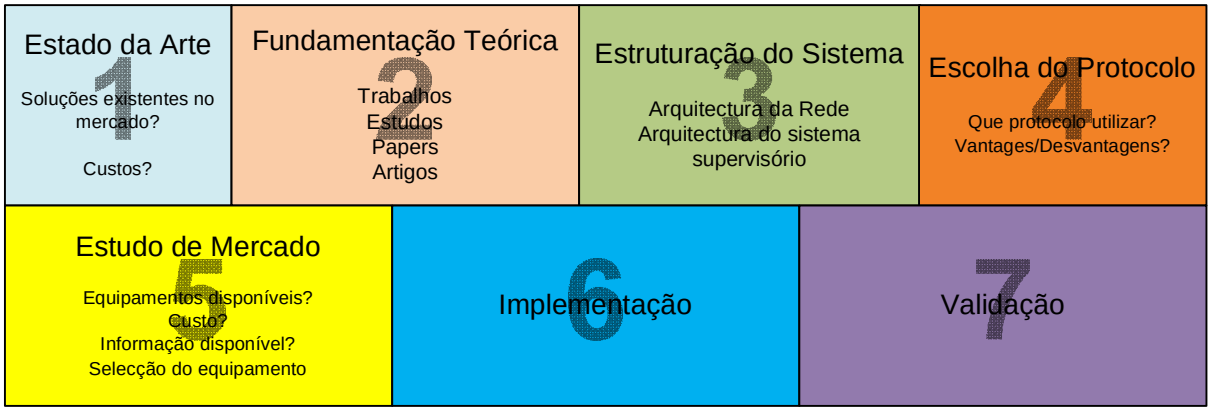


Figura 2.1 – Ciclo de intervenção

Tendo encontrado diversas soluções no mercado (eMonitor, Monitorização de Energia "By Energy Lens", H-SEM, Wi-LeM, Optimal Monitoring System, entre outros), será necessário aprofundar os conhecimentos científicos neste tema. Para tal recorreu-se à investigação de trabalhos já realizados, estudos, *papers* e artigos. Os sistemas de monitorização de consumos de energia anteriormente apresentados serão detalhados na secção 2.3.

Após o estudo teórico partiu-se para o esboço da rede e do software a implementar. Como todos os sistemas são compostos por componentes, efectuou-se um estudo de mercado com o intuito de identificar qual o material que melhor se adequava ao sistema a implementar. Aqui, vários aspectos foram levados em consideração, tais como: propriedades dos componentes, custo, assistência dada pelo fornecedor, informação disponível e claro, o orçamento disponível. No entanto, a escolha do material não poderá ser efectuada sem que primeiro seja identificado o protocolo a utilizar. Isto porque o protocolo influenciará na escolha dos componentes.

Foram estudados e avaliados vários protocolos de comunicação (CANOpen, INTERBUS-S, FIELDBUS FOUNDATION, MODBUS, STD 32, SSI, PROFITBUS, DEVICENET, entre outros) e suas vantagens e desvantagens de utilização.

CANOpen é o nome dado ao protocolo desenvolvido pelo grupo de fabricantes CAN in Automation (CIA). Este protocolo foi desenvolvido para suprimir a falta de padronização do protocolo CAN nas camadas maiores (baseando no modelo OSI). O CANOpen foi inicialmente criado para aplicações com máquinas, mas actualmente domina grande parte dos sistemas de automação, incluindo o ramo residencial.

O protocolo CANOpen pode ser resumido em três partes: A aplicação, o dicionário de objectos e o protocolo propriamente dito. A aplicação faz apenas a tradução dos dados recebidos e enviados. O dicionário de objectos contem diversos parâmetros comuns a todos os equipamentos. Esses parâmetros contêm informações sobre o estado do equipamento, estado da rede e valores específicos para os fabricantes. E, por fim, o protocolo, que é o responsável pela padronização dos dados enviados na rede [6].

Interbus-S é uma rede de trabalho e um protocolo de comunicação de alta velocidade, que permite que grandes quantidades de dados sejam transferidas rapidamente para e de nós das redes. O Baud Rate nominal para o Interbus-S é fixado em 500 Kbits/seg. As redes Interbus-S são sistemas "centralizados", precisando de um controlador mestre (normalmente um PLC) que controle todas as comunicações com nós escravos na rede de trabalho. A interface do Interbus-S é apenas um nó escravo [7].

O FIELDBUS é um protocolo desenvolvido para automação de Sistemas de Fabrico, elaborado pela FieldBus Foundation e normalizado pela ISA (The International Society for Measurement and Control). Visa a interligação de instrumentos e equipamentos, possibilitando o controle e monitoração dos processos. Geralmente é utilizado com os chamados Softwares Supervisórios (SCADA, etc.), que permitem a aquisição e visualização de dados e estados dos equipamentos.

O PROFIBUS (Process Field Bus) especifica as características técnicas e funcionais de um sistema de comunicação industrial, através do qual dispositivos digitais se podem interligar, desde o nível de campo até ao nível de células. O PROFIBUS é um sistema multi-mestre e permite a operação conjunta de diversos sistemas de automação, engenharia ou visualização, com os seus respectivos dispositivos periféricos (por ex. I/O's). O PROFIBUS diferencia os seus dispositivos entre mestres e escravos [8].

A DeviceNet classifica-se como uma rede de dispositivos, sendo utilizada para interligação de equipamentos de campo, tais como sensores, actuadores, e CLPs. Esta rede foi desenvolvida pela Allen Bradley sobre o protocolo CAN e a sua especificação é aberta e controlada pela DeviceNet Foundation [9].

Escolhido o protocolo, identificaram-se os componentes a utilizar e partiu-se para a implementação e validação/teste do sistema.

A estrutura geral do sistema a implementar é mostrada na Figura 2.2.

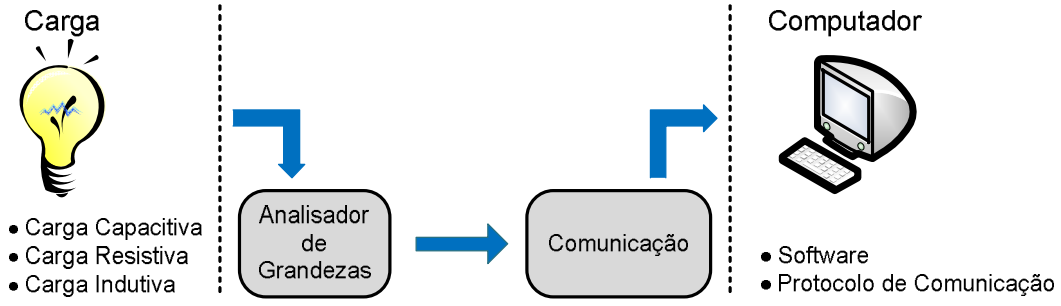


Figura 2.2 – Estrutura geral do trabalho

Baseado em cada bloco da figura anterior, é possível separar a teoria e os conceitos anteriormente apresentados nos seguintes blocos:

- Comunicação
- Sistema de supervisão

O bloco da comunicação englobará toda a parte que diz respeito aos analisadores de energia, conversores e interfaces utilizadas para que a comunicação com o sistema supervisor seja possível.

O sistema supervisor será o responsável pela recolha dos dados, armazenamento, análise e tratamento dos mesmos.

Na Figura 2.3 tem-se uma visão global de como será a solução proposta para este trabalho. O sistema supervisor será o responsável pelo armazenamento e recolha da informação. Este, estará ligado a analisadores de energia que podem estar situados em diferentes espaços físicos. A comunicação entre eles será o aspecto principal deste trabalho.

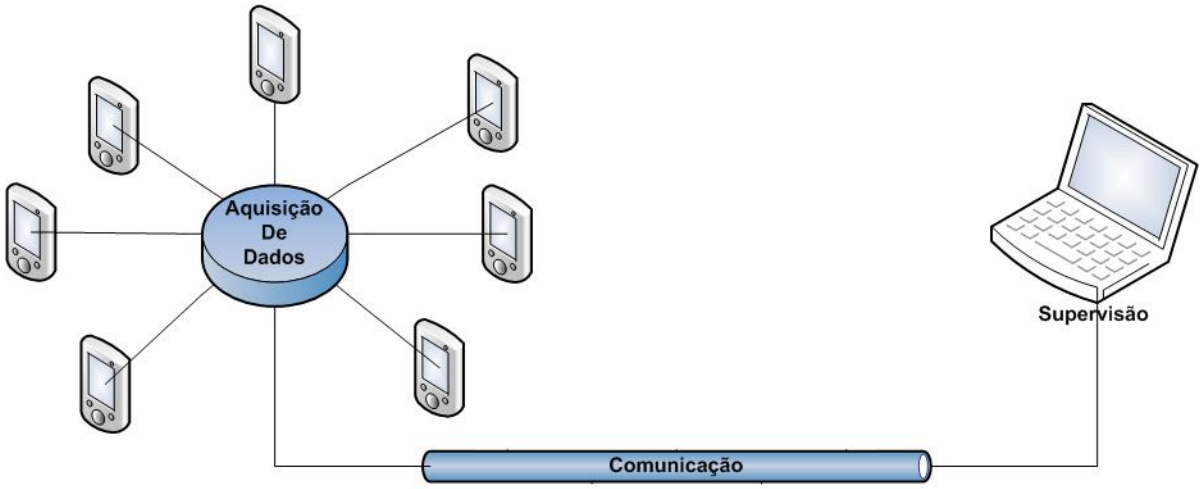


Figura 2.3 – Estrutura geral da solução

2.2 Protocolos de Comunicação – Estado da Arte

Para dois ou mais equipamentos inteligentes comunicarem entre si, torna-se necessário o uso de um protocolo de comunicação. Halsall [10] define protocolo simplesmente como um conjunto de regras para troca de mensagens. De facto, os documentos oficiais dos protocolos de comunicação definem regras para cada bit, palavra ou pacote de bytes trocados entre os equipamentos, para o correcto tráfego de informações.

Existem centenas de protocolos de comunicação para centenas de aplicações. O ideal, tanto para os fornecedores quanto para os utilizadores, é a modularidade além da padronização dos programas e dos equipamentos. Como isso é dificilmente alcançável num livre-mercado de fabricantes, existe um grande interesse em que os dispositivos inteligentes "conversem" com todos os outros, ou seja, que existam normas técnicas de aceitação universal [10].

Cada dispositivo envolvido na comunicação tem de suportar não só o mesmo protocolo como também deverá ter a mesma versão do mesmo. Quaisquer diferenças resultantes da implementação do protocolo originarão erros de comunicação.

Normalmente, quando todos os dispositivos de comunicação são do mesmo fabricante, a comunicação raramente tem erros. Assim, usando um protocolo de comunicação único, o fabricante fica numa posição privilegiada. Ou seja, qualquer dispositivo que se pretenda adquirir para comunicar com os restantes terá de possuir o mesmo protocolo e a mesma versão. Portanto, terá de ser comprado ao mesmo fabricante - o que representa um grave problema!

Com a chegada do conceito de "sistemas abertos", pretende-se que os dispositivos de um fabricante possam inter-operar com outros de um fabricante diferente. Para que tal aconteça, é necessário que um dos fabricantes utilize Protocolos Industriais Standardizados (PIS). Se isto acontecer, cada utilizador poderá comprar os dispositivos que entender, tendo em conta o preço e a finalidade a que se destinam.

A utilização de protocolos standardizados conduz à redução de preços e ao aumento da flexibilidade no que respeita à compra de dispositivos de comunicação. Assim, surgem algumas vantagens para os utilizadores:

- Independência dos fabricantes;
- Sistemas fiáveis a baixo custo;
- Especificações e conhecimento disponíveis sem custo para o utilizador.

Para os vendedores existirão também benefícios recorrentes da standardização:

- Redução dos custos de implementação e manutenção;
- Acesso a um mercado cada vez maior e, consequentemente, possibilidade de competir em termos de preços e não apenas em detalhes técnicos;
- Custo efectivo da implementação.

Todas as vantagens apresentadas têm naturalmente algum preço a pagar, nomeadamente:

- Necessidade de aumento de velocidades de transmissão para que se mantenha a mesma eficiência – relativamente ao uso do mesmo protocolo;
- Não usufruto de todas as funcionalidades de um determinado dispositivo aquando do uso de um protocolo de comunicação standardizado.

Actualmente, existem no mercado inúmeros protocolos de comunicação. Uns de acesso livre, outros apenas para venda *(Vide Anexo A)*.

Todos os protocolos apresentados encontram-se implementados nos mais diversos dispositivos no entanto, como referido, nem todos eles são de livre acesso. Assim sendo, torna-se complicado efectuar a comunicação entre dispositivos operando com diferentes protocolos.

De entre os protocolos apresentados no Anexo A, os mais utilizados em sistemas de monitorização de consumos de energia são:

- FieldBus;
- CAN;
- Ethernet;
- Modbus.

O FieldBus, foi desenhado para aparelhos de I/O remotos, com muitas funções em que a informação deve ser trocada de forma bidireccional, totalmente digital. Este protocolo permite a distribuição de aplicações de controlo através da rede, tal como a ligação com dispositivos de campo (H1 – 31,25Mbts/s) e computadores (HSE - 100Mbt/s).

O protocolo CAN é bastante utilizado na indústria automóvel e em sistemas de automação. A informação é transaccionada através de comunicação série tendo as mensagens um formato fixo. Este protocolo funciona com mecanismos de prioridades, filtragem de mensagens e possui um sistema de detecção de erros bastante fiável.

Tanto o Fieldbus como o CAN podem ser complementados com Ethernet. No entanto, a Ethernet encontra-se limitada a 100 metros e, necessita de hubs. Por outro lado, sabe-se que também não é intrinsecamente segura e, aquando da comunicação H1, existe pouca largura de banda.

O protocolo Ethernet não garante a transmissão de dados de forma determinística, o que a impede de ser uma rede tempo-real [11].

Se devidamente utilizados, os protocolos CAN e FieldBus permitem garantir restrições tempo-real devido ao mecanismo de prioridades. Tanto um como outro funcionam com cablagem bifilar, ao contrário do protocolo Ethernet. Para os primeiros não é necessária a introdução de hubs nem de routers uma vez que os dispositivos são ligados a um barramento comum.

Relativamente à Ethernet, esta permite taxas de transmissão bastante mais elevadas que o CAN/FieldBus, sendo a transmissão bastante mais eficiente nestes devido aos "*overheads*".

As tendências actuais apontam para a combinação de redes CAN/FieldBus (a nível do hardware) com Ethernet (gestão/monitorização de dados do processo).

O protocolo Modbus foi desenvolvido como um simples meio de troca de dados entre controladores e sensores, utilizando mestre-escravo / cliente-servidor. Uma vez desenvolvido, foi largamente difundido e bem aceite pelo meio industrial, e também pelo facto de se tratar de um protocolo aberto, tornou-se num dos protocolos mais implementados em diversos produtos de diversos fabricantes para a área da automação industrial.

Apesar de não ser um protocolo bastante robusto, a sua rara simplicidade permite não só uma rápida implementação mas também flexibilidade aliada a diferentes problemas industriais.

2.3 Soluções de Monitorização de Consumos de Energia

Actualmente, existem no mercado diversas soluções para monitorizar os consumos de energia, Tabela 2.1. Estas soluções permitem que o utilizador conheça de forma clara qual o seu padrão de consumo. Com esse conhecimento, torna-se mais fácil reajustar os hábitos de consumo.

Tabela 2.1 - Soluções de Monitorização de consumos de energia presentes no mercado e respectivas características

Características	H-SEM	Optimal	Wi-LeM	eMonitor	Energy Lens
Gráficos	Sim	Sim	Sim	Sim	Sim
Tabelas	Sim	Sim	Sim	Sim	Sim
Protocolos	Modbus	Modbus	Modbus	Modbus	Modbus
Impressão	Sim	Sim	Não	Sim	Sim
Email	Não	Sim	Não	Sim	Sim
Análise Online	Não	Não	Não	Não	Não

Das diversas soluções existentes no mercado, apresentam-se apenas as que estão na Tabela 2.1 pois serão, à partida, as que possibilitam monitorar consumos energéticos em grandes complexos.

O eMonitor é um sistema de monitorização que permite manter um histórico dos consumos do edifício (electricidade, gás e água), dos respectivos custos associados e das variáveis ambientais. Caracteriza-se pela flexibilidade de instalação, adaptando-se facilmente ao edifício e aos requisitos pretendidos pelo cliente. Pode ser interligado a diversos equipamentos já instalados no edifício bem como a outros existentes no mercado.

Permite definir alertas automáticos através de envio de mensagens de correio electrónico ou de SMS, informando sobre a detecção de consumos anómalos ou apenas sensibilizando os utentes do edifício. Este sistema permite o armazenamento contínuo de dados relativos aos consumos totais do edifício, desagregados por secções ou até por equipamentos [12].

O Energy Lens permite criar uma grande variedade de gráficos e tabelas dos dados de energia relativos a um intervalo de tempo [13]. Estes gráficos e tabelas podem ser utilizados para diversos tipos de análises, nomeadamente para monitorização e segmentação da energia, oferecendo uma visão global do consumo, Figura 2.4.

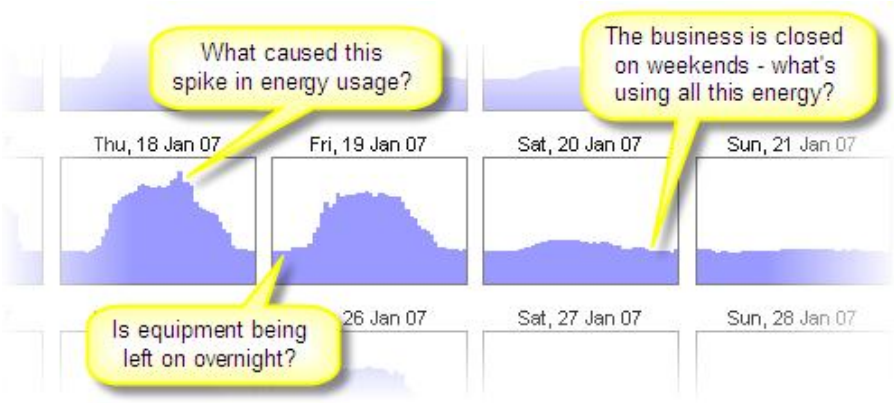


Figura 2.4 – Relatório elaborado no Energy Lens [13]

A *Hughes Energy Monitoring System* apresenta uma solução de acompanhamento e segmentação automática de energia (H-SEM), Figura 2.5. O sistema é constituído por hardware, cablagem e software [14]. O fabricante providencia a instalação, ensaios e suporte necessários para implementar o sistema. Os medidores de energia podem ser fornecidos e instalados como parte da solução ou podem ser utilizados os contadores existentes, desde que disponham de uma interface de dados compatíveis com a saída por impulsos, Modbus, M-Bus ou 4-20 mA.

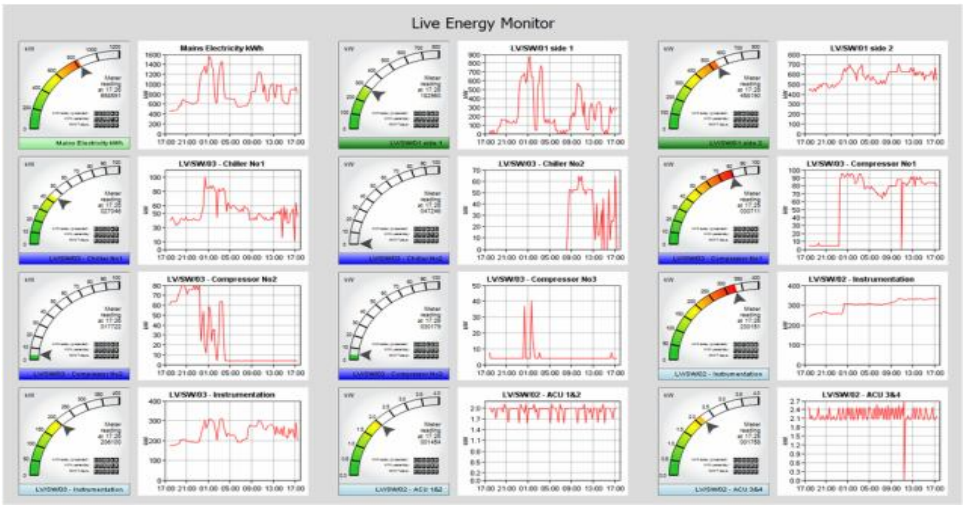


Figura 2.5 – Display do H-SEM [14]

A QEnergia oferece o Wi-LeM cuja tecnologia assenta numa topologia wireless, Figura 2.6. Trata-se de uma solução a ter em conta, no entanto, há que ter em consideração a atenuação das ondas quando as distancias entre os dispositivos são grandes ou quando existem obstáculos [15].

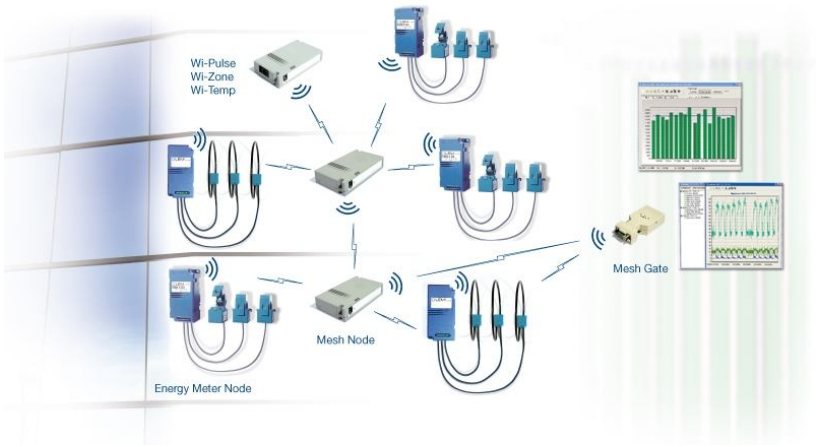


Figura 2.6 – Arquitectura do Wi-LeM [15]

O *Optimal Monitoring*, Figura 2.7, é uma solução orientada para a recolha e elaboração de relatórios de dados energéticos [16]. É utilizado para medir o consumo e os custos, podendo também enviar mensagens de alarme em caso de falha de alimentação ou simples inactividade do equipamento, mas também caso detecte consumos anómalos.

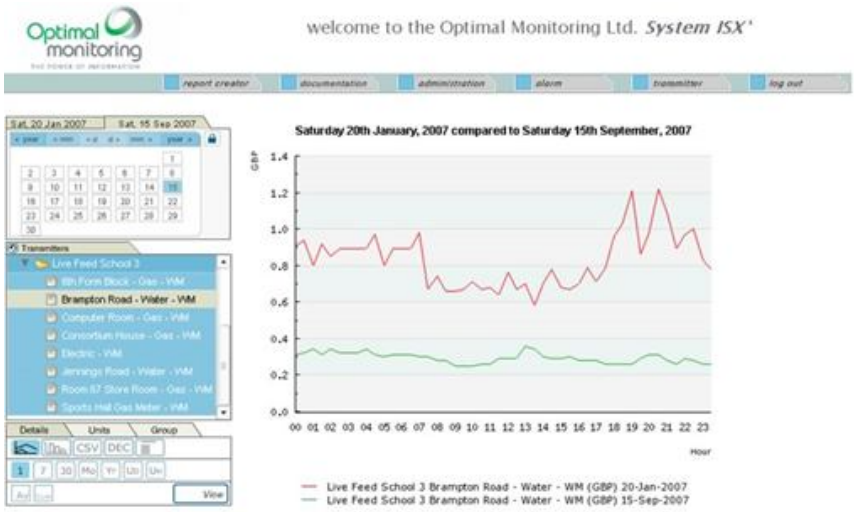


Figura 2.7 – Layout do Optimal Monitoring [16]

Pela Tabela 2.1, pode-se verificar que as soluções presentes no mercado (neste caso decidiu-se mostrar as mais importantes) utilizam sempre o protocolo Modbus, seja ele como protocolo principal ou não.

O projecto a desenvolver pretende ser uma alternativa às soluções disponíveis no mercado que prime pelo reduzido custo mantendo a fiabilidade e funcionalidade em níveis bastante elevados.

No capítulo seguinte será dado ênfase ao protocolo Modbus que, foi o escolhido para a implementação deste projecto. Serão ainda apresentadas as razões que levaram à sua escolha em detrimento de outros protocolo de comunicação.

Capítulo 3 -
Protocolo de Comunicação Modbus

A Automação industrial consiste no conjunto de operações automáticas controladas mediante componentes mecânicos, electrónicos, programas de computadores e quaisquer combinações destes, que substituem o controlo e as decisões humanas. Para comunicarem, os dispositivos deverão utilizar o mesmo protocolo de comunicação. Neste capítulo serão abordados os protocolos de comunicação industrial, nomeadamente o Modbus e os seus diferentes modos de funcionamento. No final, apresenta-se o modelo adoptado para a implementação da solução proposta.

3.1 Histórico da Automação Industrial

É importante frisar a importância que a comunicação de dados assume numa arquitectura distribuída. A comunicação é responsável pelo dimensionamento e modularidade dos sistemas que, não se tem conseguido com grande sucesso devido à grande quantidade de protocolos de comunicação privados existentes no mercado. Estes protocolos, associados a diversas aplicações, não obedecem a padrões de domínio público.

A interoperabilidade é um dos requisitos importantes, por possibilitar a dois ou mais sistemas trocarem informações, gerando dessa forma, uma certa independência de um determinado fornecedor. Para que esse requisito seja verdadeiro, é necessário o estabelecimento de padrões de protocolos de comunicação, os quais são compostos pelas interfaces e meios de conexão entre os sistemas a nível físico (interface eléctrica e mecânica) e a nível lógico (protocolo de acesso ao meio, protocolo de rede, protocolo de transporte, protocolo de sessão e protocolo de aplicação).

A utilização de um protocolo de comunicação baseado em normas ou padrões de domínio público vem a possibilitar a independência de um determinado aplicativo.

Actualmente, devido ao grande avanço tecnológico, as redes de automação são largamente utilizadas, apresentando vantagens relativamente a sistema convencionais de cabos: diminuição da cablagem, facilidade na manutenção, flexibilidade na configuração da rede e, principalmente, diagnóstico dos dispositivos. Além disso, por usarem protocolos de comunicação digitais padronizados, essas redes possibilitam a integração de equipamentos de vários fabricantes distintos. Tais sistemas dizem-se abertos, e são uma tendência em todas as áreas da tecnologia devido à sua flexibilidade e capacidade de expansão.

Os protocolos de comunicação foram inicialmente desenvolvidos para sistemas usados em automação industrial que, remonta à década de 20, com a criação das linhas de montagem de automóveis por Henry Ford, Figura 3.1. Daí para cá, o avanço tecnológico nas mais diversas áreas da automação industrial, tem sido cada vez maior, proporcionando um aumento de qualidade e quantidade de produção, bem como uma redução dos custos inerentes.

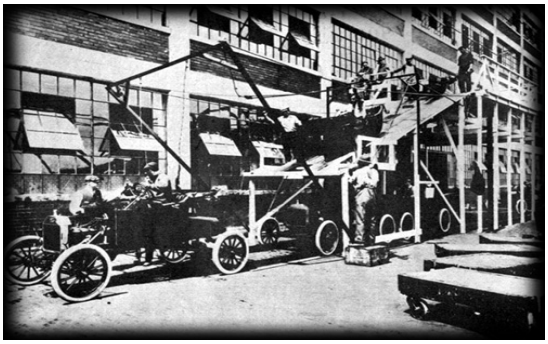


Figura 3.1 - Linha de Montagem do Ford "T" Fonte: Enciclopédia Britânica

A automação é a substituição do trabalho humano, ou animal, por máquina com a mínima interferência do operador humano. Trata-se portanto do controlo de processos

automáticos. Por automático, entenda-se ter um mecanismo de actuação própria, que realize uma determinada acção num tempo determinado ou como resposta a uma certa condição.

O conceito de automação está sujeito ao ambiente e à experiência da pessoa envolvida. São exemplos de automação:

- Para uma dona de casa: Máquina de lavar roupa;
- Para um empregado da indústria automóvel: Robô
- Para uma pessoa comum: Capacidade de tirar dinheiro do multibanco.

Associado ao conceito de automação está a ideia de usar potência eléctrica ou mecânica para accionar algum tipo de máquina. Assim sendo, a automação deverá acrescentar à máquina algum padrão de inteligência para que ela possa executar uma determinada tarefa de modo mais eficiente e com vantagens económicas e de segurança [17].

O progresso da automação decorre dos avanços tecnológicos, provenientes da electrónica e da informática, com o aparecimento do CLP (Controlador Lógico Programável) como equipamento fundamental para a automação e os programas de computador que tornaram possível a sua interligação.

A automação pode assim diminuir a mão-de-obra empregada porém, ainda requer operadores. Em vez de fazer a tarefa directamente, o operador controla a máquina que executa a tarefa. No entanto, é necessário que o operador seja formado para poder trabalhar com ela.

Para muitas pessoas, automação é sinónimo de perda de empregos. Porém, o contrário também pode acontecer. De facto, a falta de automação emprega muitas pessoas. No entanto, essas empresas não podem competir economicamente com outras devido à baixa produtividade, sendo depois forçadas a despedir pessoas ou até a encerrar actividade.

Muitas aplicações de automação não requerem a substituição de pessoas porque a função ainda não existia antes ou é impossível de ser feita sem ser manualmente.

A automação está intimamente ligada à instrumentação. Existem inúmeros instrumentos empregues para a realização da automação.

Historicamente, o primeiro termo usado foi o de controlo automático de processo. Foram usados instrumentos com as funções de medir, transmitir, comparar e actuar no processo, para se conseguir o produto desejado com pequena ou nenhuma ajuda humana – controlo automático.

Com o aumento da complexidade dos processos, exigências de produtividade, segurança e protecção do ambiente, além do controlo automático do processo, apareceu a necessidade de monitorar o controlo automático.

A partir deste novo nível de instrumentos, com funções de monitorização e alarme é que apareceu o termo automação. Assim sendo, apareceram funções bastante importantes, tais como: detecção, comparação, alarme e actuação lógica [17]. Funções estas bastante importantes quando aplicadas no campo da monitorização distribuída, como por exemplo a monitorização de consumos de energia.

A monitorização distribuída de sistemas é um desafio cada vez maior. Maior ainda se os objectos de monitorização se encontrarem fisicamente distantes, em edifícios diferentes ou, se se encontrarem em redes privadas sem que exista a possibilidade de se efectuarem ligações directas entre os diferentes componentes da arquitectura de monitorização.

Para que os diferentes dispositivos de uma arquitectura de monitorização distribuída possam comunicar, há que estabelecer um protocolo, tal como é mencionado na secção seguinte.

3.2 Protocolos de Comunicação Industrial

Os sistemas de automação são constituídos por, actuadores, controladores e sensores, interfaces homem máquina e sistemas de supervisão que possibilitam a interacção entre o sistema e o operador. Para que o processo a automatizar possa ser executado e monitorado correctamente é necessário que, os elementos constituintes desse sistema comuniquem entre si de forma clara e inequívoca. Quem proporciona este tipo de comportamento são os protocolos de comunicação.

Sem protocolos, uma rede de comunicação, qualquer que seja, não funciona. Os protocolos determinam a forma como um programa deve preparar os dados, para serem enviados ao nível seguinte do processo de comunicação. Desta forma, os protocolos de comunicação usados nas redes podem ser comparados a idiomas, ou linguagens, que servem para estabelecer a comunicação entre os equipamentos electrónicos e as máquinas na indústria.

Uma analogia pode ajudar a explicar a ideia de uma comunicação em vários níveis usando protocolos. Imagine-se: dois filósofos (dois processos pares) um dos quais fala urdu e inglês (1) e o outro (2) fala chinês e francês. Como não falam um idioma comum, contratam tradutores que, por sua vez, têm uma secretária cada. O filósofo 1 deseja transmitir o seu gosto por *oryctolagus cuniculus* ao seu par. Para tal, ele envia uma mensagem (em inglês) ao seu tradutor, na qual diz *"I like rabbits"*, como mostra a Figura 3.2. Como os tradutores resolveram usar um idioma neutro, o holandês, a mensagem foi convertida para *"Ik vind konijnen leuk"*. - A escolha do idioma é o protocolo -.

O tradutor entrega a mensagem a uma secretária para ser transmitida, por exemplo, pelo fax (protocolo da camada 1). Quando chega, a mensagem é traduzida para o francês e passada para o filósofo 2. Assim, cada protocolo é totalmente independente dos demais, desde que as interfaces não sejam alteradas. Nada impede que, os tradutores mudem do

holandês para o finlandês, desde que, ambos concordem com a modificação. De modo semelhante, as secretárias também podem passar de fax para correio electrónico ou telefone [18].

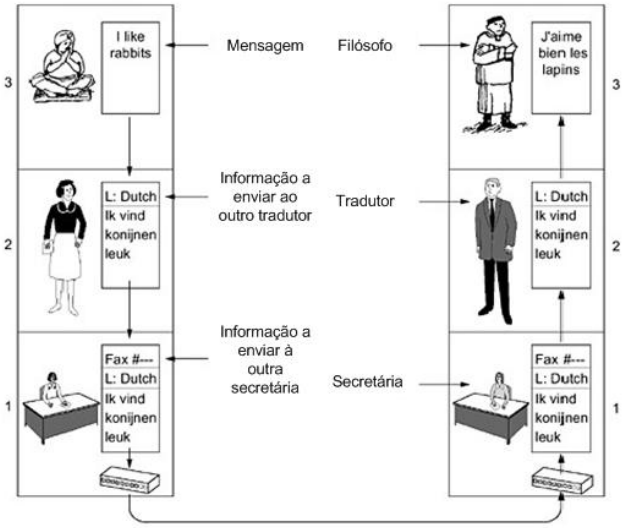


Figura 3.2 - A arquitectura filósofo-tradutor-secretária [18]

Desta forma, cada tipo de equipamento utiliza um determinado protocolo de comunicação e, por sua vez, cada protocolo utiliza um meio físico diferente para a transmissão dos sinais de comunicação.

Alguns protocolos de comunicação e suas características podem ser observados na Tabela 3.1. (No Anexo A consta uma lista bem mais extensa de protocolos de comunicação).

Tabela 3.1 – Características de alguns protocolos de comunicação

Protocolo	Originalmente usado por	Velocidade	Princípio de acesso	Camadas OSI
MODBUS	Gould-Modicon	19.2 kbps	Consulta cíclica	1,2,7
SPABUS	ABB (exclusivo)	19.2 kbps	Consulta cíclica	1,2,7
DNP 3.0	GE-Harris	19.2 kbps	Consulta cíclica	1,2,7[3]
IEC 60870-5	Todos	19.2 kbps	Consulta cíclica	1,2,7
MODBUS+	Gould-Modicon		Token	1,2,7
PROFIBUS	Siemens	1.2 Mbps	Token	1,2,7
MVB	ABB	1.5 Mbps	TDM	1,2,7[4]
FIP	Merlin-Gerin	2.5 Mbps	TDM	1,2,7
Ethernet + TCP/IP	Todos	10 Mbps	CSMA/CD	1 - 7
LON	ABB	1.25 Mbps	PCSMA/CD	1 - 7
UCA 2.0	GE	10 Mbps	CSMA/CD	1 - 7

No ponto seguinte veremos como funciona o protocolo Modbus, qual a forma das suas mensagens, modos de transmissão e como é recolhida a informação de um dispositivo operando com este protocolo.

3.3 O Protocolo Modbus

3.3.1 Descrição Geral

O protocolo Modbus foi criado em 1979 pela Modicon Industrial Automation Systems (actual Shneider Electric) com o objectivo de integrar CLP's, computadores, terminais, sensores e outros dispositivos de controlo e monitorização. Trata-se de um protocolo de mensagens posicionado na sétima camada do modelo de referência OSI (Open Systems Interconnection), Figura 3.3 - Protocolo Modbus e Modelo de Referência OSI, que provê a comunicação "Mestre/Escravo" entre dispositivos ligados a diferentes tipos de barramentos ou topologias de rede [19].

Nível	Modelo OSI	
7	Aplicação	Protocolo Modbus
6	Apresentação	Vazio
5	Sessão	Vazio
4	Transporte	Vazio
3	Rede	Vazio
2	Dados	Modbus em linha série
1	Físico	EIA/TIA-485 ou EIA/TIA-232

Figura 3.3 - Protocolo Modbus e Modelo de Referência OSI [19]

O Modbus interage com diferentes tipos de redes, tais como RS232, RS485 e TCP possibilitando a comunicação através de redes Ethernet ou até mesmo da Internet. O uso de gateways permite a comunicação entre os vários barramentos/redes, Figura 3.4. As gateways são nada mais nada menos que conversores de protocolo que actuam entre redes distintas. Podem funcionar como clientes ou como servidores.

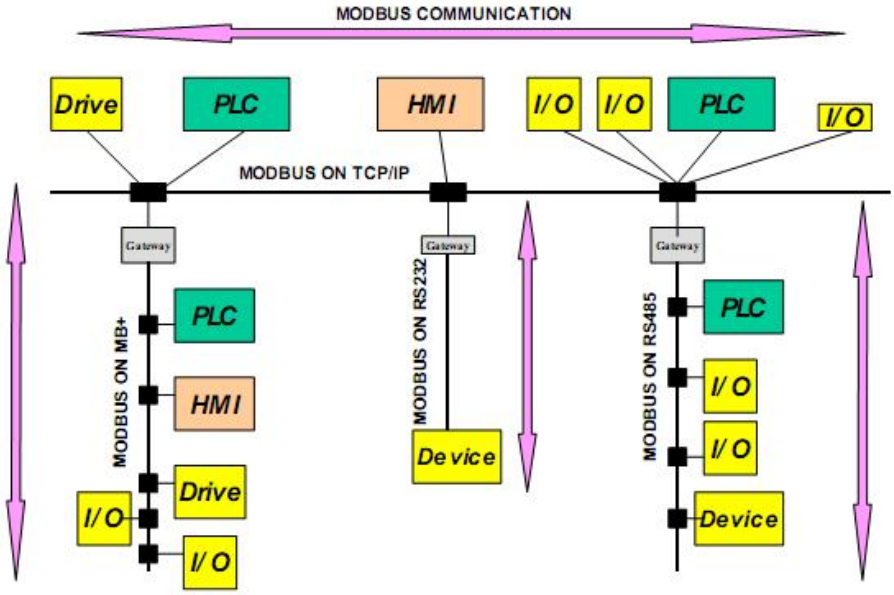


Figura 3.4 – Exemplo de uma rede Modbus [20]

Como referido, o Modbus é um protocolo de comunicação baseado no modelo "Mestre/Escravo" (M/E) onde, o "Mestre" controla todas as actividades da linha série

através de sondagens selectivas a cada um dos escravos a ela associados. Na mesma linha, a cada "Mestre" podem estar conectados no máximo 247 "Escravos" identificados por endereços diferentes.

Logicamente que podem ser endereçados mais de 250 dispositivos. No entanto, os transceptores[1] de uma rede RS-485 não suportam, fisicamente, essa quantidade. O protocolo Modbus estabelece um limite de 32 dispositivos. Apenas se esses dispositivos possuírem transceptores de baixa carga é que será possível colocar mais dispositivos na mesma linha.

Algumas das características deste protocolo são fixas, tais como: o formato das mensagens, a sequência, a verificação dos erros e as funções que executa. Outras características podem ser seleccionadas pelo utilizador, como por exemplo, a escolha do meio de transmissão, o baud rate, a paridade, o número de stop bits e os modos de transmissão (ASCII ou RTU) - estes parâmetros não podem ser alterados com o programa em execução.

O Modbus não é o protocolo de comunicação mais poderoso, porém, a sua simplicidade, aliada à facilidade de implementação tornam-no num dos protocolos mais utilizados em automação industrial [21]. A Shneider Electric decidiu disponibilizar este protocolo a toda a comunidade. Desta forma, foi criado o website *www.Modbus-IDA.org* [22] - em conjunto com um grupo de fabricantes - onde é possível fazer o download do código fonte do protocolo Modbus bem como obter ajuda na resolução de questões relacionadas com o protocolo. Actualmente, este site pertence a uma comunidade composta por todas as empresas e entidades que, de alguma forma, estão ligadas a este protocolo.

3.3.2 Mapa de Memória de Dispositivos Modbus

Cada dispositivo Modbus possui memória onde as variáveis contendo informação são armazenadas. A especificação Modbus determina o modo e o tipo de dados que podem ser recolhidos. No entanto, não existe uma especificação de como e onde o fornecedor dos dispositivos deve colocar essa mesma informação no mapa de dados.

O protocolo especifica apenas que devem existir quatro zonas de dados, distintas, dentro do mapa de informação, Tabela 3.2:

- Discrete Inputs;
- Coils (Outputs);
- Input Registers (Input Data);
- Holding Registers.

[1] Dispositivo que combina um transmissor e um receptor utilizando componentes de circuito comuns para ambas funções num só aparelho.

Tabela 3.2 – Possível mapa de dados de um dispositivo Modbus [22]

Table Addresses	Type	Table Name
1-9999	Read or Write	Coils
10001-19999	Read Only	Discrete Inputs
30001-39999	Read Only	Input Registers
40001-49999	Read or Write	Holding Registers

Esta é a principal desvantagem deste protocolo pois cada fornecedor pode desenhar o seu mapa de informação e, desta forma, decidir onde começa e acaba cada zona de dados. Se houver uma rede com muitos dispositivos diferentes, o programador ou instalador, deverá ter em atenção este pormenor pois o endereçamento será diferente de dispositivo para dispositivo. Para contornar este problema, o fabricante publica o mapa de informação junto com o dispositivo.

As entradas discretas e as *coils* possuem valores de 1 bit apenas e têm endereços específicos. As entradas analógicas são armazenadas em registos de 16 bits. Utilizando dois destes registos, o protocolo Modbus suporta o padrão IEEE de 32 bits com vírgula flutuante. Os *Holding Registers* são também de 16 bits e suportam vírgula flutuante.

3.3.3 Modbus em Linha Série

O protocolo Modbus em linha série é um protocolo mestre-escravo. Isto significa que existe apenas um mestre (dispositivo) simultaneamente ligado ao barramento. Quanto aos escravos, dependendo da interface física que se esteja a utilizar, um ou mais podem estar simultaneamente ligados ao mesmo barramento.

Ao nível físico, o Modbus pode usar vários tipos de interfaces RS232 e RS485. Em RS232, como é uma comunicação ponto a ponto, apenas existe um mestre e um escravo. No caso da RS485 é possível ter vários escravos mas, apenas um único mestre. Normalmente, um mestre é um sistema supervisor enquanto os escravos são controladores lógicos programáveis.

A comunicação é sempre iniciada pelo mestre. Os escravos nunca transmitem dados a menos que isso lhes seja solicitado. Para o trabalho a implementar, isto é um aspecto importante pois não se pretende que os dispositivos estejam constantemente a enviar informação ao mestre. Cabe a este interrogar cada escravo, quando necessitar de alguma informação. Por outro lado, a comunicação entre escravos é inexistente. O mestre inicia somente uma comunicação de cada vez, Figura 3.5 [23].

Este protocolo – Modbus -, sendo simples, permite manipular facilmente os escravos, daí não haver necessidade de recorrer a protocolos mais complexos.

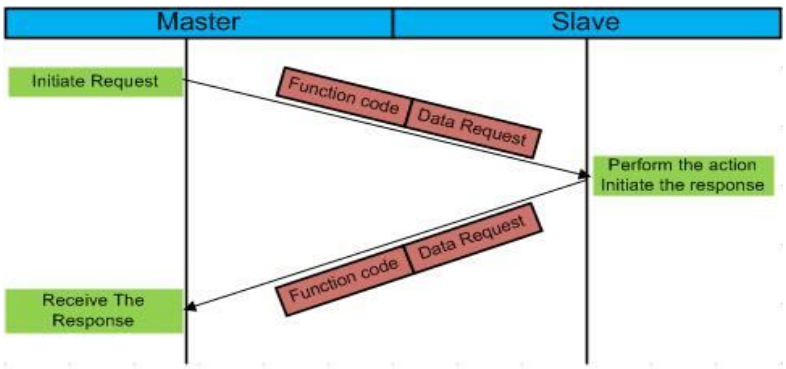


Figura 3.5 – Comunicação mestre-escravo

Um mestre pode solicitar informação de qualquer escravo utilizando uma de duas vias:

- *Unicast* – O mestre envia a mensagem a um determinado escravo. Após recebida, a mensagem é tratada e uma resposta é enviada ao mestre, Figura 3.6.

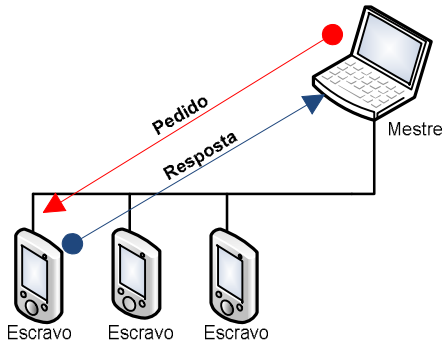


Figura 3.6 – Modo Unicast [23]

- *Broadcast* – O mestre envia uma mensagem a todos os escravos. Estes, ao receberem a mensagem, executam-na mas, não é enviada nenhuma resposta ao mestre, Figura 3.7.

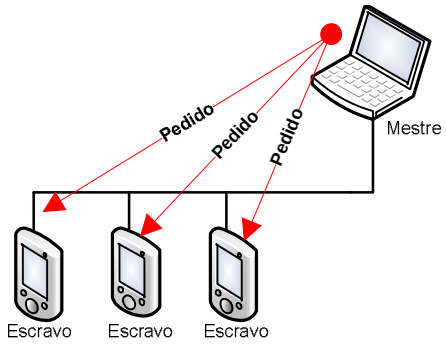


Figura 3.7 – Modo Broadcast [23]

Sabendo quais os modos de transmissão entre Mestre e Escravo, é altura de verificar como funciona o ciclo de pergunta – resposta em dispositivos que usam o protocolo Modbus.

3.3.4 O Ciclo Pergunta-Resposta

Na mensagem de consulta, o código da função informa o escravo, com o respectivo endereço, qual a acção a executar. Os bytes de dados contêm a informação necessária para que o escravo saiba, por exemplo, qual o registo onde deve começar a leitura e quantos registos deve ler. O campo da verificação de erros permite ao escravo efectuar a validação dos dados recebidos, Figura 3.8 [24].

Na mensagem de resposta repete-se o código da função. Os bytes destinados aos dados contêm os dados requisitados pelo mestre ou então o estado em que o escravo se encontra. Se ocorre um erro, o código da função é alterado para indicar que a resposta é uma resposta de erro e, os bytes de dados trarão um código que descreve esse mesmo erro.

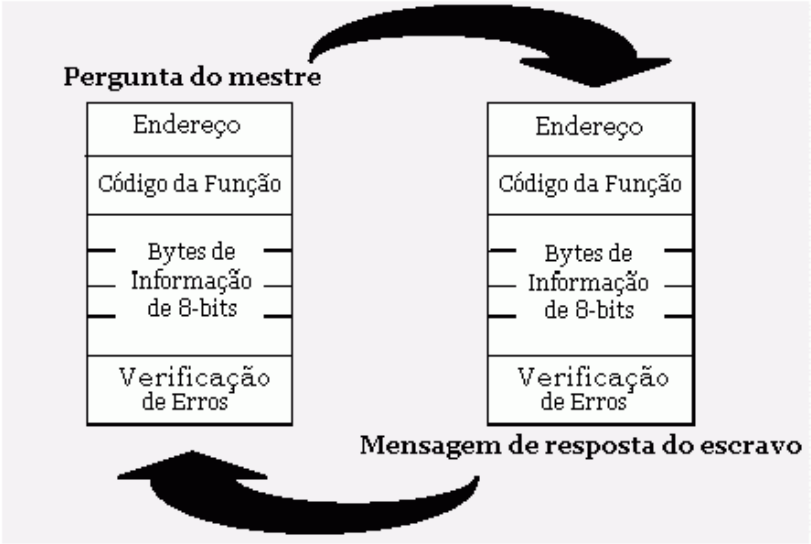


Figura 3.8 – Ciclo Pergunta-Resposta [24]

Posto isto, é necessário conhecer o formato das mensagens Modbus, como é construída cada trama e qual o conteúdo de cada campo da mesma.

3.3.5 Formato da Mensagem Modbus

O protocolo Modbus define um PDU (*Protocol Data Unit*), Figura 3.9, independente do meio físico a utilizar.

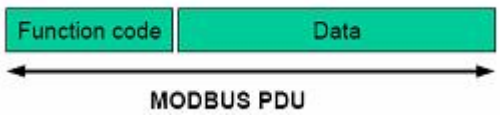


Figura 3.9 – Protocol Data Unit (PDU) em Modbus

A construção da mensagem do protocolo Modbus num determinado meio físico é realizada através da introdução de campos adicionais ao PDU. Assim, o mestre, para iniciar a comunicação com um dado escravo, constrói a mensagem adicionando ao PDU um campo

com o endereço do escravo com o qual pretende comunicar e um campo destinado à verificação de erros, Figura 3.10 [24].

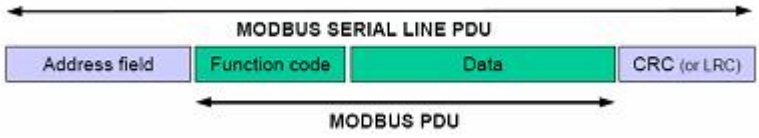


Figura 3.10 – Frame Modbus usada em linha série

Os campos da mensagem acima referida são semelhantes tanto quando esta é enviada pelo mestre como quando é enviada pelo escravo, Tabela 3.3.

Tabela 3.3 - Campos da mensagem Modbus enviada pelo Mestre e pelo Escravo

Frame	Enviada pelo mestre	Enviada pelo escravo
Endereço	Contém o endereço do escravo com o qual se pretende comunicar.	Contém o endereço do próprio escravo para que o mestre saiba qual a origem da mensagem.
Código da Função	Função que o mestre pretende que o escravo execute.	Quando não ocorrem erros de execução, este campo contém informação igual à enviada pelo mestre. Se o escravo não consegue executar a tarefa, envia um código de erro.
Dados	Contém informação complementar ao campo anterior.	Contém a informação pedida pelo mestre ou um código de erro.
Verificação de Erros	Usado para verificar erros na frame.	Usado para verificar erros na frame.

Desta forma, a detecção do erro e a identificação dos escravos fica bastante simplificada.

Como se acabou de verificar, o Modbus define um determinado PDU e, a ele são acrescentados novos campos, originando assim a mensagem Modbus. Tendo a mensagem construída, é altura de transmitir. Os modos de transmissão podem ser ASCII ou RTU.

3.3.6 Modos de Transmissão

A transmissão em linha série usando o protocolo Modbus pode seguir dois modos: modo RTU (*Remote Terminal Unit*) e modo ASCII (*American Standard Code for Information Interchange*). Não é permitido usar os dois modos de transmissão na mesma rede.

Na comunicação Modbus em RTU, quando em linha série, cada byte (8 bits) da mensagem é enviado através de 2 caracteres hexadecimais, isto é, uma "palavra" (*"word"*),

Figura 3.11. Cada mensagem deve ser transmitida numa sequência de *"words"*. A principal vantagem deste modo de transmissão deve-se ao facto de a sua grande densidade de caracteres permitir um processamento de dados mais eficaz do que o modo ASCII (para o mesmo baud rate), aumentando assim o desempenho da comunicação. Neste modo, a verificação de erros é baseada no algoritmo CRC-16.

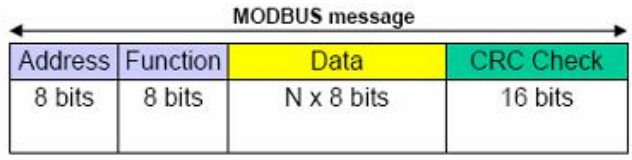


Figura 3.11 – Formato de uma frame Modbus em RTU [24]

Por sua vez, no modo ASCII, cada byte de uma mensagem é enviado como dois caracteres ASCII, Figura 3.12. Este modo é menos eficiente que o modo RTU, segundo [24], uma vez que para cada byte são necessários dois caracteres. No modo ASCII a verificação de erros assenta no LRC (Longitudinal Redundancy Check).

Start	Address	Function	Data	LRC	End
1 char :	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

Figura 3.12 – Formato de uma frame Modbus em ASCII [24]

Este modo de transmissão é especialmente recomendado quando os equipamentos ou a linha não se adequarem à especificação do modo RTU no que diz respeito à gestão dos intervalos de tempo necessários à transmissão e recepção [24].

Quando o mestre envia uma mensagem, coloca-a num quadro, Figura 3.13, definindo inequivocamente o início e o fim da mesma. Isto permite aos dispositivos receptores que recebam um novo quadro, conhecerem o início da mensagem e quando esta é totalmente recebida. Ao serem detectadas mensagens parciais, geram-se códigos de erro que, são enviados ao mestre como resposta.

No modo RTU, os quadros de mensagem são separados por intervalos de silêncio de pelo menos 3,5 tempos de caracter.

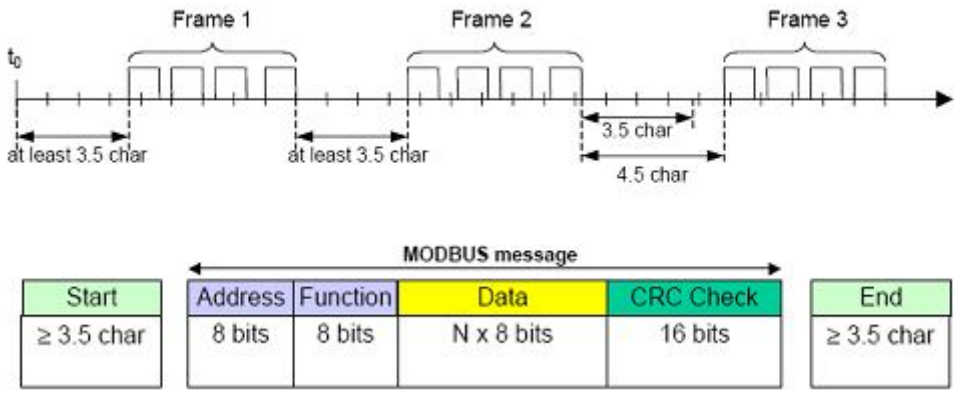


Figura 3.13 – Quadro de mensagem em modo RTU [24]

O quadro inteiro da mensagem deve ser transmitido com um fluxo constante de caracteres. Se um tempo de silêncio maior do que 1,5 tempos de caracter (definido internamente) for detectado o quadro da mensagem é declarado incompleto e deve ser descartado pelo receptor, Figura 3.14.

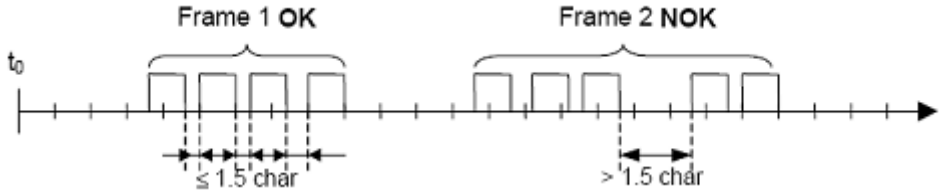


Figura 3.14 – Rejeição do quadro de mensagem [24]

No que toca aos tempos de comunicação mestre/escravo, Figura 3.15, existem alguns aspectos a ter em consideração:

- A duração das fases de requisição, resposta e *broadcast* depende das características da comunicação (comprimento da mensagem e processamento);
- A duração das fases de espera e tratamento dependem do tempo de processamento da requisição necessário para a aplicação contida nos nós escravos.

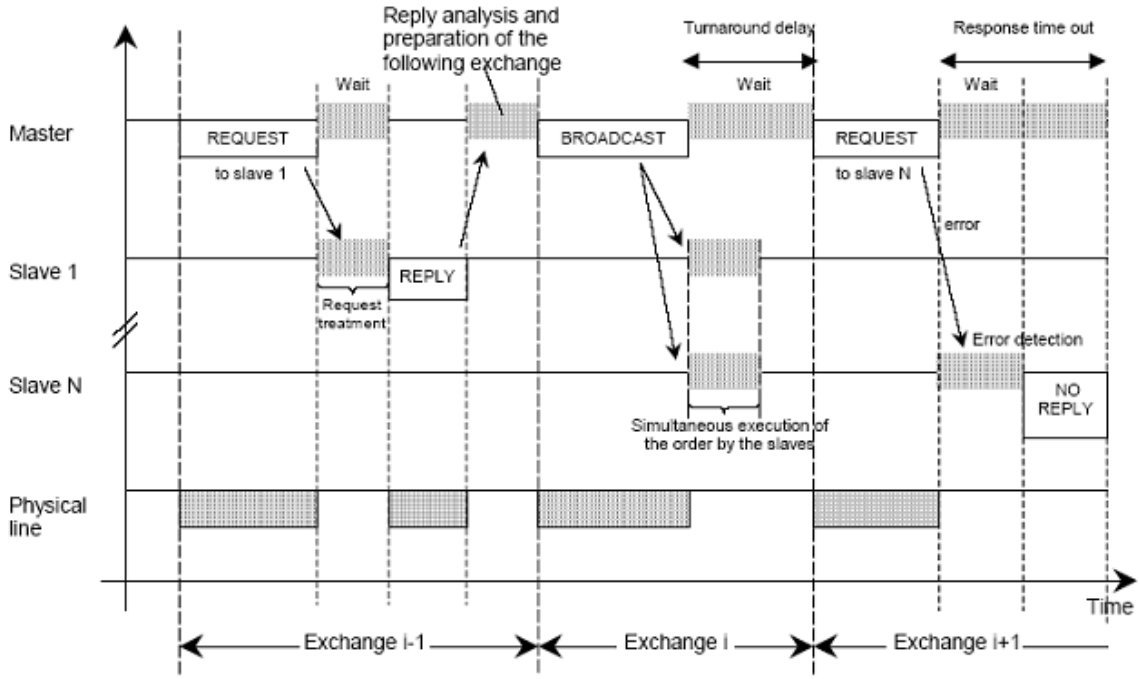


Figura 3.15 – Diagrama de tempos de comunicação Mestre/Escravo [24]

Visto o diagrama de tempo de comunicação Mestre/Escravo, com que rapidez se poderão efectuar os pedidos/ordens?

Pois bem, primeiro há que relembrar que se continua a trabalhar com uma ligação série e, portanto, não será possível efectuar pedidos mais depressa do que o que seria possível através de uma linha série. De facto, se forem adicionados muitos sistemas de pedidos *(queuing systems)* entre a aplicação e os dispositivos, perder-se-á algum desempenho. Por exemplo, alguns testes de desempenho mostraram que o download remoto por ModbusTCP para Modbus RTU é mais lento que o download directo para ModbusRTU em cerca de 20 por cento [25].

Acima de tudo, há que relembrar que a velocidade da linha série (*baud rate*) é responsável pela maior parte do consumo de tempo - Tabela 3.4. Suponhamos um pedido de leitura de 125 registos. Isto requer uma resposta de 255 bytes que, a 19.2kbps necessita de mais de 133ms apenas para percorrer toda a parte física da ligação, a 300 baud, leva aproximadamente 10 segundos! No caso da solução que vem sendo apresentada, conseguem-se intervalos de tempo de leitura de 1 segundo entre amostras, o que demonstra a robustez da comunicação implementada. Isto deve-se às pequenas dimensões das linhas onde estão acoplados os analisadores de energia (escravos).

Tabela 3.4 – Conversão do Baud Rate para diferentes unidades [25]

Baud Rate	Byte/Sec	Bit Time (msec)	Byte Time (msec)	256 Byte Time (msec)	(in sec)
300	30	3.333333	33.333333	8533.333333	8.53
600	60	1.666667	16.666667	4266.666667	4.27
1200	120	0.833333	8.333333	2133.333333	2.13
2400	240	0.416667	4.166667	1066.666667	1.07
4800	480	0.208333	2.083333	533.333333	0.53
9600	960	0.104167	1.041667	266.666667	0.27
19200	1920	0.052083	0.520833	133.333333	0.13
38400	3840	0.026042	0.260417	66.666667	0.07
57600	5760	0.017361	0.173611	44.444444	0.04
115200	11520	0.008681	0.086806	22.222222	0.02

O tempo total necessário para se efectuar uma leitura corresponde à soma dos seguintes intervalos de tempo:

- Atraso inerente ao reconhecimento entre o Mestre e o Escravo de um pedido de leitura;
- Atraso inerente ao TCP/IP sobre Ethernet;
- Atraso para o pedido de leitura atravessar a rede Ethernet e chegar ao destino;
- Espera que a linha série esteja livre;
- Atraso no escravo para poder reconhecer, processar e responder um pedido;
- Atraso correspondente ao tempo de deslocamento da resposta.

Com o evoluir da tecnologia, pretende-se que os sistemas sejam cada vez mais céleres, tanto no processamento, como na transmissão de dados. Assim, quando a rapidez é um factor a ter em conta, deve-se usar o protocolo Modbus aliado ao TCP.

A aplicação implementada consegue trabalhar sobre TCP, o que em grandes complexos se revela uma mais-valia pois não há necessidade de passar cablagem. Basta ao utilizador aproveitar as ligações de rede já estabelecidas no complexo industrial. Isto reduz consideravelmente os custos de instalação.

3.3.7 Modbus sobre TCP/IP

Actualmente, requerem-se sistemas cada vez mais rápidos e versáteis. De forma a aumentar o desempenho do protocolo Modbus, foi desenvolvido uma nova variante que utiliza a Ethernet como meio físico. Assim nasceu o protocolo Modbus TCP/IP que, acrescenta à simplicidade do Modbus as vantagens do TCP/IP sobre Ethernet.

Se a rede Ethernet possuir ligação à Internet é possível fazer qualquer tarefa em qualquer ponto do mundo como se se estivesse no mesmo edifício do equipamento. A

acrescentar a todas estas vantagens, o IEC (International Electrotechnical Commission) [26] integrou o Modbus TCP/IP nas normas IEC61158 e IEC61784-2.

O protocolo Modbus define um PDU independente do meio físico a utilizar, Figura 3.16.

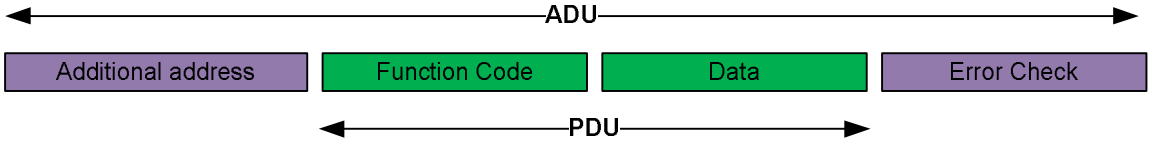


Figura 3.16 – Mensagem genérica em Modbus [26]

A construção da mensagem do protocolo Modbus em determinado meio físico é realizada com a introdução de campos adicionais ao PDU. O cliente que inicia a comunicação constrói o PDU, ao qual adiciona os campos necessários à transmissão da mensagem no meio em questão.

Em Modbus TCP/IP um campo específico é acrescentado ao PDU do Modbus, Figura 3.17. Este campo tem o nome de "MBAP Header" (Modbus Application Protocol Header). Em Modbus TCP/IP não é utilizada a verificação de erros na frame, de igual forma ao CRC do Modbus RTU ou LRC do Modbus ASCII. O Modbus TCP/IP deixa a verificação de erros na frame a cargo dos protocolos TCP/IP e Ethernet.

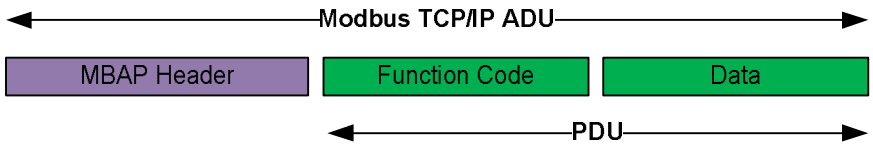


Figura 3.17 – Application Data Unit (ADU) em Modbus sobre TCP/IP [26]

Este *header* (cabeçalho) contém informações diferentes quando comparadas com o ADU utilizado no Modbus RTU em linha série:

- Sobre TCP, no MBAP, é incluída informação do comprimento da frame para que o receptor possa identificar os limites da mensagem mesmo que esta tenha sido dividida em vários pacotes.
- O endereço do escravo é substituído por um *unit identifier* no cabeçalho MBAP;
- Todos os pedidos e respostas são construídos de tal forma que o receptor consiga verificar que a mensagem terminou. Para os *function codes* onde o PDU tem um comprimento fixo, apenas o *function code* é suficiente. Para *function codes* que necessitam de enviar dados adicionais no *request* ou na *response*, o campo data inclui um byte de contagem.

No Modbus TCP/IP qualquer equipamento pode ser cliente e servidor simultaneamente. Aqui é possível ter mais do que um mestre. Na configuração Ethernet com a utilização de bridges, routers e gateways é possível, no mesmo sistema, ter equipamentos ligados em Ethernet e outros ligados em linha série. Qualquer mestre ligado na rede

Ethernet pode comunicar com outro equipamento ligado em qualquer local do sistema esteja ele ligado na Ethernet ou ligado num barramento série que por sua vez estará ligado à Ethernet através de uma gateway.

Uma comunicação em Modbus TCP/IP requer o estabelecimento de uma ligação TCP entre o cliente e o servidor. Esta ligação pode ser realizada pelo módulo de aplicação ou pelo módulo de gestão de ligações TCP. No primeiro caso é necessário que a aplicação utilize recursos especiais para a gestão das ligações, enquanto que no segundo caso todo o processo de ligações é realizado automaticamente. Desta forma, o utilizador apenas se concentra no envio e recepção de mensagens em Modbus.

O número de escravos e mestres numa rede Ethernet depende das características dos equipamentos. Numa rede Ethernet é possível haver mais do que uma transacção em simultâneo. É ainda possível, no caso de dois equipamentos terem características de escravo e de mestre, haver duas comunicações em curso simultaneamente entre estes dois equipamentos. O equipamento terá a porta TCP 502 para troca de mensagens no servidor e uma outra porta qualquer para troca de mensagens no cliente. A porta TCP 502 está reservada para comunicações Modbus TCP/IP.

Em Modbus TCP/IP um cliente pode estabelecer outras comunicações sem a primeira estar terminada. O controlo do fluxo de mensagens é gerido pelos protocolos TCP/IP e Ethernet. Uma trama TCP apenas transporta uma trama Modbus de cada vez. Não é possível encapsular dois Modbus PDU numa trama TCP.

3.4 Modelo Modbus Adoptado

Com o desenvolvimento de novos sistemas, aplicações e protocolos, a simplicidade do Modbus aliada à sua facilidade de implementação sobre os meios de comunicação existentes continua a permitir que este seja o protocolo mais apoiado e implementado a nível industrial.

Quando os utilizadores de sistemas de controlo mais antigos se deparam com a necessidade de expandir a instrumentação de campo já existente ou com a adição de controladores remotos, frequentemente optam pela utilização do Modbus por este oferecer uma solução simples para problemas complexos. Além disso, quando há a necessidade de interligar um dispositivo a um sistema de controlo, usar a interface Modbus do dispositivo revela-se, frequentemente, o método mais fácil.

Embora Modbus seja um dos métodos de comunicação mais antigos, é também a nível industrial, o mais utilizado pois é fácil de usar, confiável, de baixo custo, e interliga-se a quase todos os dispositivos de detecção e controlo.

Pelas razões agora apresentadas e por outras mais que veremos mais à frente, o protocolo Modbus foi o escolhido para a implementação deste trabalho.

Existem dois modos de transmissão em comunicação série usando Modbus - ASCII e RTU. Estes modos, como já foi referido, determinam a forma como as mensagens são codificadas em Modbus.

No modo ASCII, as mensagens são codificadas usando valores hexadecimais. Assim, para cada byte de informação, são usados dois bytes pois todas as comunicações byte apenas definem 4 bits no sistema hexadecimal. Já no Modbus RTU a comunicação é feita em binário onde cada byte de dados é codificado num só byte de comunicação.

No formato ASCII, as mensagens são lidas, enquanto no RTU as mensagens são em código binário e, não podem ser lidas durante a monitorização – são processadas em *background*. Em contrapartida, as mensagens RTU possuem menor tamanho, o que permite uma maior troca de dados no mesmo espaço de tempo. Este aspecto é particularmente importante em ambientes de monitorização pois consegue-se monitorar uma grandeza em pequenos intervalos de tempo (1 segundo).

Existem algumas limitações relativamente ao Modbus RTU. Por exemplo, a arquitectura cliente/servidor requer que para que os dados cheguem ao cliente, o nó do cliente deve consultar o do servidor. Outra limitação é que o Modbus RTU pode ser multiponto, ou seja, podem operar tantos dispositivos servidores numa rede Modbus RTU, quanto a camada física esteja preparada para suportar. Por forma a permitir múltiplas "redes escravas" é necessário a introdução de um dispositivo RS-485 como por exemplo um modem.

Com essa configuração, é possível que vários nós sejam multiponto, mesmo quando se usam baixas taxas de transmissão. Segundos ou até minutos podem passar antes que as mudanças possam ser efectuadas nos dispositivos de campo. Isto é devido à baixa velocidade e elevada quantidade de dispositivos/nós Modbus envolvidos.

Se a rapidez relativa à transferência de dados é imperativa, deve-se usar um protocolo diferente do Modbus RTU, como por exemplo o Ethernet/IP ou o BACnet.

Apesar das limitações do Modbus RTU, existem inúmeras razões que justificam o facto de este continuar a ser o mais implementado em automação industrial. Por um lado, é muito mais fácil de implementar do que os protocolos mais recentes e é uma força dominante no mercado industrial. Por outro lado, exige muito menos memória. O Modbus RTU pode facilmente ser inserido em 2kb de memória de um pequeno CPU (Central Processing Unit) de 8 bits ou então num PIC[2] (Programmable Interface Controller). Se por acaso fosse usado outro protocolo, como o BACnet ou Ethernet/IP, seriam necessários entre 30 e 100kb de memória [27].

O Modbus TCP/IP usa o TCP/IP e a Ethernet para fazer circular a informação entre os diferentes dispositivos. Ou seja, o Modbus TCP/IP combina a rede física (Ethernet), com o

[2] Os PIC são uma família de microcontroladores fabricados pela Microchip Technology, que processam dados de 8 bits, de 16 bits e, mais recentemente, de 32 bits.

protocolo TCP/IP, e um standard de representação de dados que é o Modbus. Isto é bastante importante para o problema em análise, como veremos adiante, pois pretende-se trabalhar com grandes complexos.

A solução TCP/IP pode, no entanto, não ser a melhor para algumas aplicações que exigem tempos de resposta não compatíveis com os fornecidos pelo TCP/IP.

Tal como referido anteriormente, o Modbus RTU é facilmente encapsulado com TCP/IP. Tratando-se de um protocolo simples, robusto e pouco complexo, resolveu-se escolhê-lo em detrimento de outros, para a implementação deste trabalho.

3.4.1 Conteúdo das mensagens Modbus

Escolhido o protocolo a utilizar, é chegada a altura de estudar o conteúdo das mensagens Modbus de forma a responder a algumas questões:

- Quais as funções disponíveis no protocolo Modbus?
- Qual o conteúdo da mensagem enviada pelo Mestre?
- O que acontece se a mensagem não for entregue ao Escravo?
- Como é descodificada a mensagem de resposta?
- O que fazer com os dados recebidos?

Os únicos identificadores através dos quais o dispositivo mestre pode reconhecer a resposta para uma determinada mensagem são o endereço do escravo e a função solicitada. Assim, o envio de múltiplas requisições, em que tais parâmetros coincidam, deve ser feito ordenadamente – cada mensagem só deve ser enviada após a recepção da resposta à mensagem anterior.

Podem ser enviados simultaneamente comandos iguais para dispositivos diferentes ou comandos diferentes para um mesmo dispositivo, embora neste ultimo caso, possam surgir problemas dependendo do equipamento específico [28].

A Figura 3.18 mostra um exemplo de uma mensagem Modbus enviada pelo Mestre enquanto a Figura 3.19 apresenta a resposta. Ambos os exemplos mostram como a mensagem seria codificada, tanto em ASCII como em RTU.

A mensagem enviada pelo Mestre consiste num pedido de leitura (*Read Holding Registers*) de registos ao Escravo com o endereço 06. Pretende-se saber o que existe nos registos 40108 até 40110.

QUERY

Field Name	Example (Hex)	ASCII Characters	RTU 8–Bit Field
Header		: (colon)	None
Slave Address	06	0 6	0000 0110
Function	03	0 3	0000 0011
Starting Address Hi	00	0 0	0000 0000
Starting Address Lo	6B	6 B	0110 1011
No. of Registers Hi	00	0 0	0000 0000
No. of Registers Lo	03	0 3	0000 0011
Error Check		LRC (2 chars.)	CRC (16 bits)
Trailer		CR LF	None
	Total Bytes:	17	8

Figura 3.18 – Mensagem enviada pelo Mestre (em RTU e ASCII) [28]

O escravo, em resposta ao mestre, ecoa o código da função indicando que é uma resposta normal – não houve erro de transmissão. O campo *'Byte Count'* especifica quantos itens de 8 bits serão retornados.

RESPONSE

Field Name	Example (Hex)	ASCII Characters	RTU 8–Bit Field
Header		: (colon)	None
Slave Address	06	0 6	0000 0110
Function	03	0 3	0000 0011
Byte Count	06	0 6	0000 0110
Data Hi	02	0 2	0000 0010
Data Lo	2B	2 B	0010 1011
Data Hi	00	0 0	0000 0000
Data Lo	00	0 0	0000 0000
Data Hi	00	0 0	0000 0000
Data Lo	63	6 3	0110 0011
Error Check		LRC (2 chars.)	CRC (16 bits)
Trailer		CR LF	None
	Total Bytes:	23	11

Figura 3.19 – Mensagem de Resposta enviada pelo Escravo (em RTU e ASCII) [28]

O exemplo anterior introduziu um novo conceito – Funções. As funções ou *function codes* indicam ao Escravo as funções a realizar. São válidos códigos entre 1 e 255 (os códigos entre 128 e 255 estão reservados para funções de *exception response* – funções de resposta para indicar erros na mensagem recebida).

Existem algumas funções às quais é possível acrescentar sub-funções para determinadas tarefas.

No protocolo Modbus existem três tipos de funções, Figura 3.20:

- *Public function codes*. Funções definidas e garantidas como únicas. São validadas pela comunidade Modbus-IDA. Trata-se de funções standard implementadas em todos os equipamentos que utilizam o protocolo Modbus.
- *User-defined function codes*. Funções definidas pelo utilizador. Estas funções só estão disponíveis nos equipamentos onde o utilizador as instalar.
- *Reserved function codes*. Funções utilizadas por alguns fabricantes. Não estão disponíveis para o utilizador (daí também não serem mostradas na Figura 3.20.

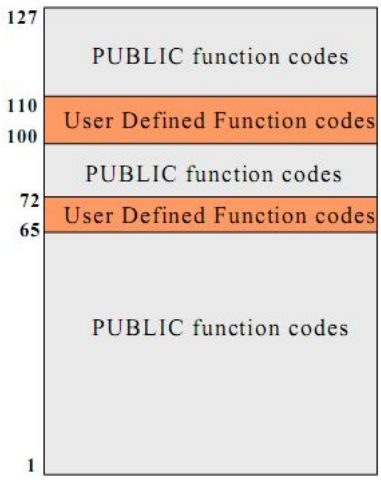


Figura 3.20 – Categorias das funções Modbus [28]

São 19 as *public function code* que podem ser utilizadas por um Mestre para solicitar tarefas a um Escravo, Figura 3.21:

- *Read coils (0x01)* - Utilizada para ler o estado de saídas (outputs);
- *Read discrete Inputs (0x02)* - Utilizada para ler o estado de entradas (inputs).
- *Read holdings registers (0x03)* - Utilizada para ler os valores dos *Holding Registers*.
- *Read input registers (0x04)* - Utilizada para ler os valores dos input registers.
- *Write single coil (0x05)* - Força uma saída (output).
- *Write single register (0x06)* - Esta função é utilizada para forçar um valor de holding register.
- *Read exception status (0x07)* - Utilizada para obter informação de status.
- *Diagnostics (0x08)* - Esta função é utilizada para realizar testes á porta de comunicação.
- *Get comm event counter (0x0B)* - Informações de status.
- *Get comm event log (0x0C)* - Esta função é utilizada para obter várias informações de status.
- *Write multiple coils (0x0F)* - Utilizada para forçar várias saídas (outputs).
- *Write multiple registers (0x10)* - Esta função é utilizada para forçar o valor de vários *Holding Registers*.
- *Report slave ID (0x11)* - Obter o ID e várias informações de status.
- *Read file record (0x14)* - Ler vários registos.
- *Write file record (0x15)* - Gravar vários registos.
- *Mask write register (0x16)* - Esta função é utilizada para fazer alterações nos registos utilizando funções lógicas.
- *Read/write multiple registers (0x17)* - Permite executar uma tarefa de escrita e de leitura com o mesmo *request*.
- *Read FIFO queue (0x18)* - Esta função permite ler informações do conteúdo da tabela de FIFO.
- *Encapsulated interface transport (0x2B)* - Função reservada.

- *CANopen general reference request and response PDU (0x2B / 0x0D)* - Esta função serve para encapsular serviços de forma a permitir a troca de informações com um sistema CAN [28].
- *Read device identification (0x2B / 0x0E)* - Permite obter informações acerca do equipamento.

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Slave ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19

Figura 3.21 - Definição das *Public Function Code* [28]

Além das *function codes* atrás enumeradas, pode-se definir outras na gama ainda não ocupada. Todas as *function codes* implementadas pelo utilizador devem ser programadas em todos os equipamentos ligados ao barramento ou rede. Só assim será possível que todos os equipamentos possam interpretar as mensagens a ele destinadas.

Quando um Mestre faz um pedido a um Escravo, o Mestre espera uma resposta normal. Uma de quatro situações pode ocorrer nesta fase:

- Se o Escravo recebe um *request* sem erros de comunicação e pode tratá-lo normalmente, então, envia ao Mestre uma resposta normal, Figura 3.22.

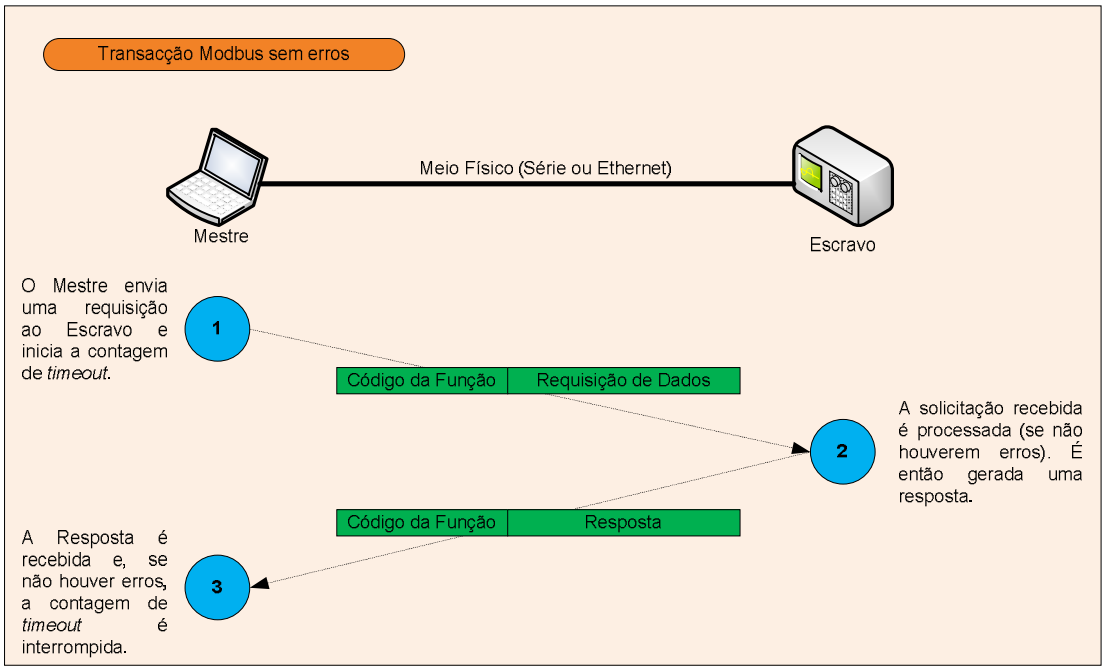


Figura 3.22 – Transacção Modbus sem erros (Adaptado da especificação técnica Modbus [29])

- Se o Escravo não recebe um determinado *request* então, também não envia qualquer resposta. Assim, passado algum tempo, o Mestre verifica que o pedido não foi tratado – note-se que o tempo de espera á definido pelo Mestre - , Figura 3.23.

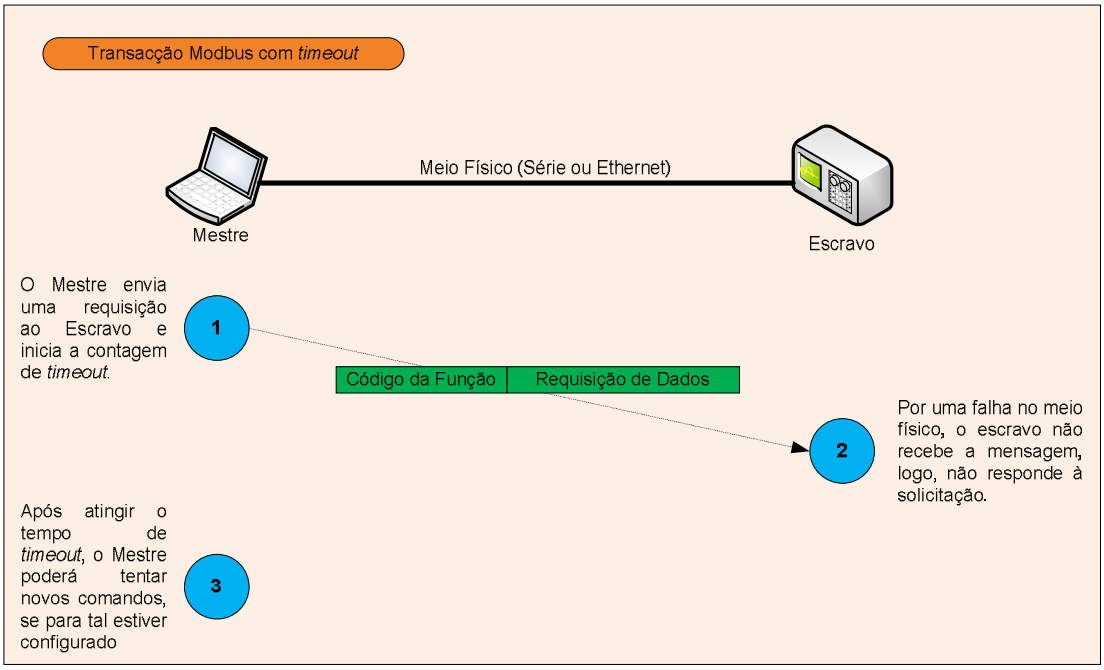


Figura 3.23 – Transacção Modbus com *timeout* (Adaptado da especificação técnica Modbus [29])

- Se o Escravo recebe um *request* e detecta um erro de comunicação, não será enviada qualquer resposta. Desta forma, ao fim de algum tempo, o Mestre verificará que o pedido não foi executado.

- Se o Escravo recebe um pedido e não é detectado qualquer erro de comunicação mas, não consegue executar a tarefa, então é enviada uma resposta (*exception response*) ao Mestre com informação desse erro, Figura 3.24.

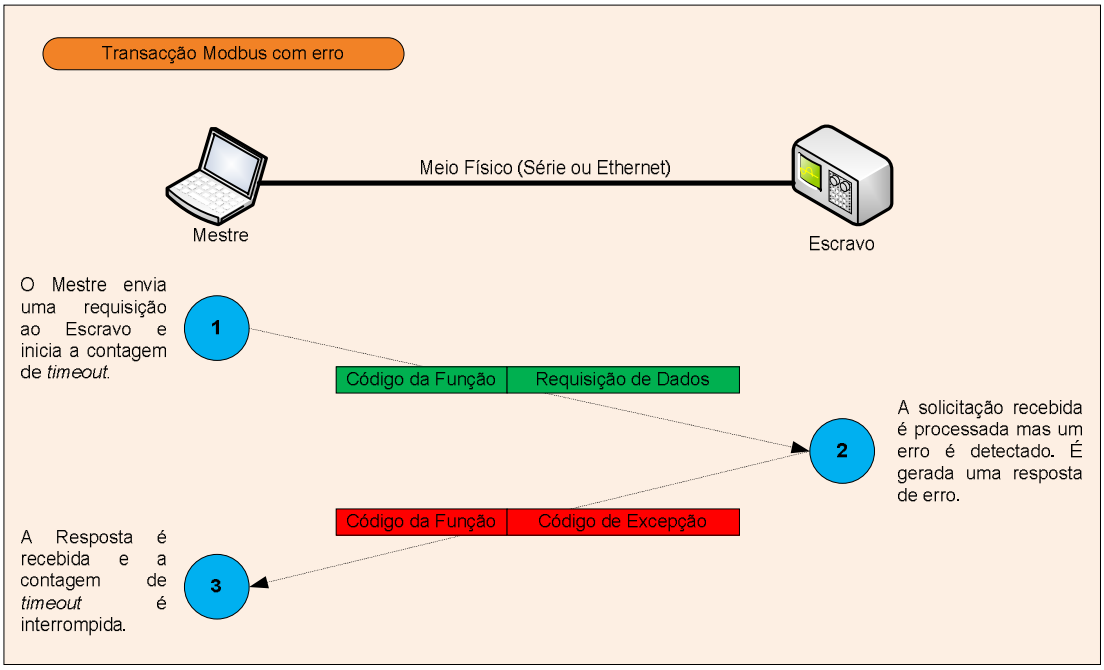


Figura 3.24 – Transacção Modbus com erros (Adaptado da especificação técnica Modbus [29])

As mensagens de *exception response* têm dois campos diferentes de uma resposta normal.

No campo *function code*, numa resposta normal, o Mestre envia o mesmo *function code* do pedido no *function code* da resposta. Como todas as *function codes* têm o bit mais significativo (MSB) a 0 (zero), nas mensagens de *exception response*, o Escravo coloca esse mesmo bit a 1 (um). Assim, as *function codes* de pedido e as *function codes* das *exception responses* apenas diferem no MSB, o que equivale a dizer que o *function code* das *exception responses* é incrementado de 128, ou 80 HEX.

Na Figura 3.25 é mostrado um diagrama de comunicação entre um Mestre e um Escravo. Na parte superior da observa-se uma comunicação sem erros e, na parte inferior, uma comunicação com erros.

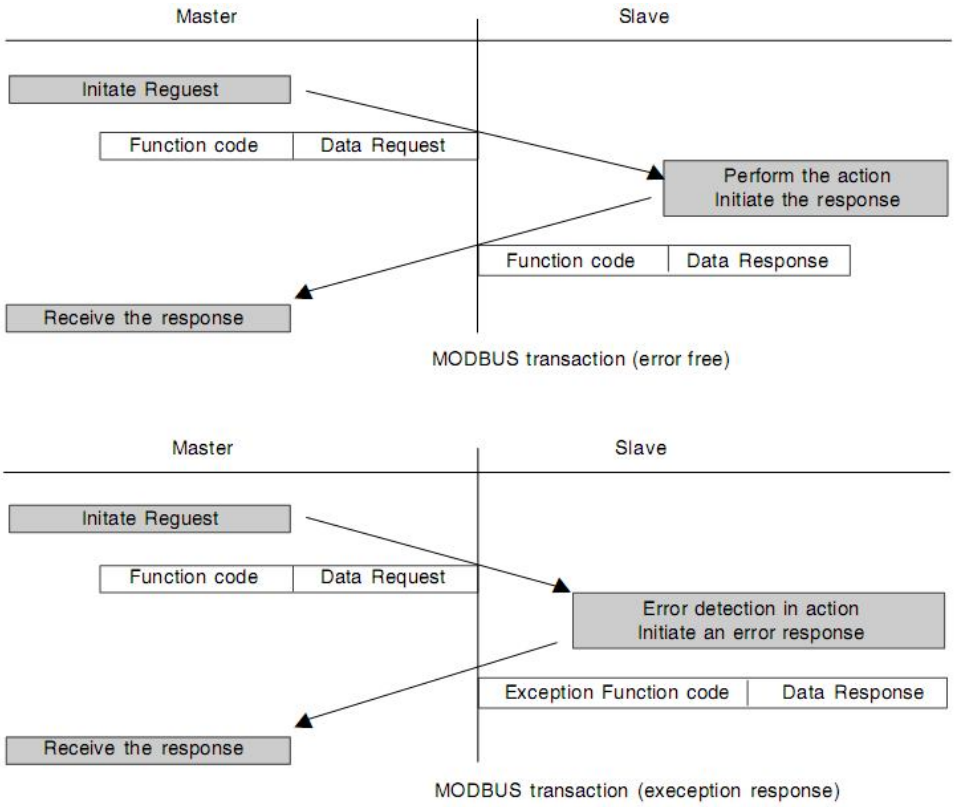


Figura 3.25 – Comunicação Modbus com e sem erros [30]

Desta forma, o Mestre ao receber uma *function code* com o MSB a 1 (um), saberá que não se trata de uma resposta normal e, por conseguinte, irá procurar no campo "*Data*" o respectivo código de erro. Os códigos de erro são conhecidos *à priori* tanto pele Mestre como pelo Escravo [30], Tabela 3.5. No Anexo B apresenta-se uma tabela completa com os códigos de excepção e os seus significados.

Tabela 3.5 – Modbus Exception Codes [30]

Modbus Exception Codes	
Code	Name
01	Illegal Function
02	Illegal Data Address
03	Illegal Data Value
04	Slave Device Failure
05	Acknowledge
06	Slave Device Busy
08	Memory Parity Error
0A	Gateway Path Unavailable
0B	Gateway Target Device Failed To Respond

Na Tabela 3.6, o Mestre envia um pedido de leitura das saídas ao Escravo (*function code* 01). Neste caso, é pedido o valor da saída com o endereço 1185 (04A1 hex) e, o número de registos a ler é 1 (0001).

Tabela 3.6 – Transmissão com ocorrência de um erro [30]

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	01	Function	81
Starting Address Hi	04	Exception Code	02
Starting Address Lo	A1		
Quantity of Outputs Hi	00		
Quantity of Outputs Lo	01		

Se o Escravo não possui o endereço indicado no pedido do Mestre então, é enviada uma resposta com o *function code* incrementado de 80 hex (01+80=81 hex) mais o código do erro. Neste caso, é o código 02 que indica *illegal data address*.

A Figura 3.26 ilustra o fluxograma da função apresentada no exemplo anterior.

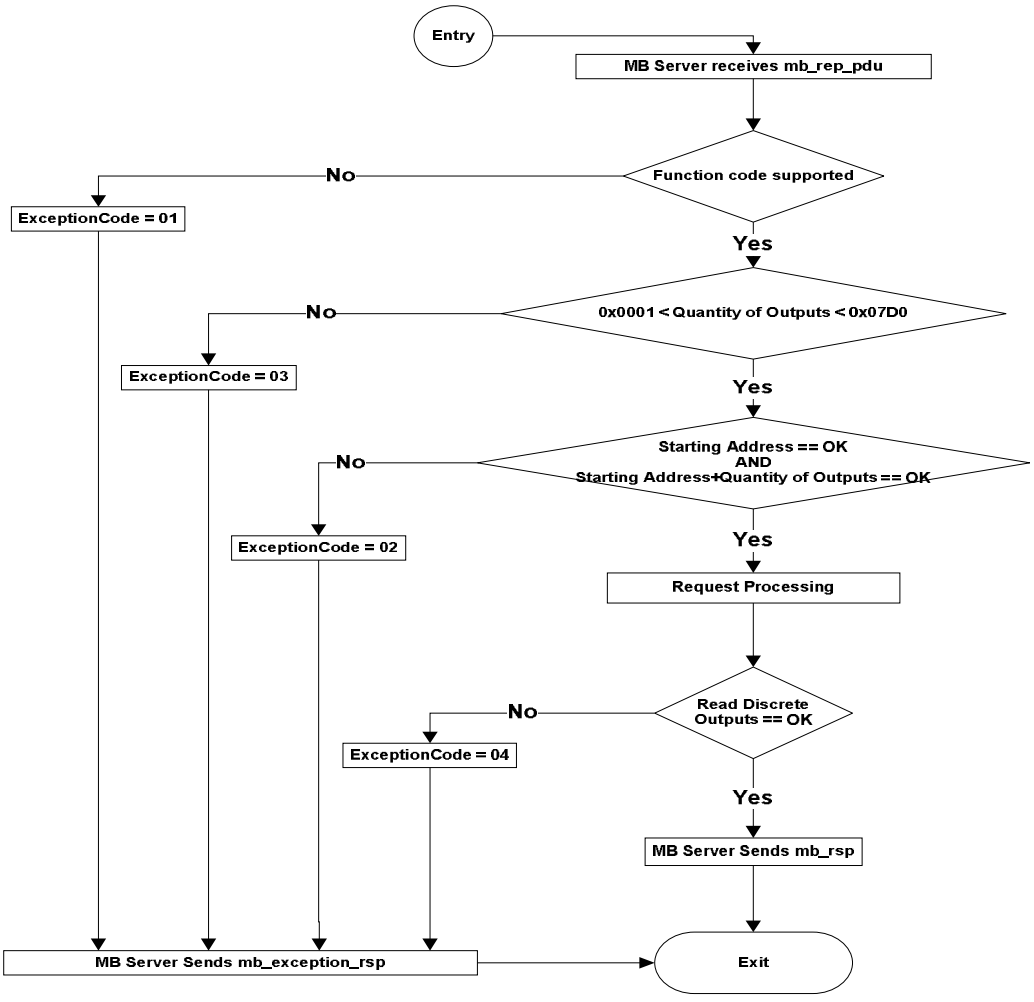


Figura 3.26 – Fluxograma da função 01

3.4.2 Dados em Tempo Real

A preservação do meio ambiente e a racionalização dos consumos de energia é o grande desafio deste início de século.

A utilização de energias renováveis e os estudos nesta área têm vindo a crescer significativamente a partir da década de 70 com a crise do petróleo e respectiva preocupação com as reservas finitas desta matéria-prima [31].

As directivas da Comunidade Europeia apontam para uma redução do consumo energético, o que conduzirá a um aumento da eficiência energética de 6 por cento em 2012. De modo a cumprir as metas relativamente ao consumo de energia, seria útil que os consumidores tivessem acesso a informações relativamente ao seu padrão de consumo e respectivo diagrama de carga.

Actualmente, existem no mercado algumas soluções para o problema apresentado porém, quando saímos do plano doméstico para o plano industrial, deparamo-nos com outro problema – o custo [32]. Se a nível doméstico, o número de pontos de controlo é reduzido, já a nível industrial ou em grandes complexos, o número de pontos de medida aumenta rapidamente tornando essas soluções demasiadamente dispendiosas.

Tendo em conta o referido no parágrafo anterior, desenvolveu-se uma aplicação - *open source* - sobretudo dedicada a grandes complexos. A aplicação em causa possui duas variantes:

- Análise em tempo real
- Recolha e armazenamento de dados, taxação e análise

A Figura 3.27 mostra a primeira variante do software desenvolvido – análise em tempo real. Esta aplicação permite visualizar os valores das correntes, tensões e potências, enfim, as principais características de uma rede eléctrica (neste caso trata-se de uma rede trifásica). Existe também uma parte para a configuração do sistema que, será observada mais em pormenor no Capítulo 4 – Implementação.

Os valores medidos podem ser usados na construção de diagramas de carga que permitam uma melhor compreensão do consumo energético. Os dados são guardados em ficheiros *.xls* (*Microsoft Office Excel*) que podem simplesmente ser armazenados no próprio computador ou enviados para uma conta de e-mail de um possível responsável que, efectuará a sua análise.

Não é possível ter mais do que um Mestre numa rede Modbus RTU. Portanto, se a gateway for configurada como Mestre, só poderá existir uma gateway. Não podem ser usadas múltiplas gateways para ler vários registos do mesmo escravo Modbus ao mesmo tempo. No entanto, podem existir múltiplas gateways numa rede Modbus RTU a partir do momento que sejam configuradas como escravas.

Se se estiver a usar dispositivos RS-232, só podem existir no máximo dois dispositivos independentemente da forma como estejam configurados. Isto acontece pois o RS-232 não é multiponto.

Na análise em tempo real, a aplicação só pode efectuar uma ligação em simultâneo a um dispositivo porque o protocolo requer a reserva de uma porta de comunicação (no computador) para cada dispositivo.

Tal faz sentido uma vez que se pretendem efectuar análises em tempo real. Ou seja, se existissem mais dispositivos simultaneamente conectados, tendo em conta o protocolo, a aplicação demoraria algum tempo a varre-los a todos. Assim, originar-se-ia um *delay* que teria de ser levado muito em conta aquando da análise do consumo energético, distorcendo-se também o conceito de *real-time*.

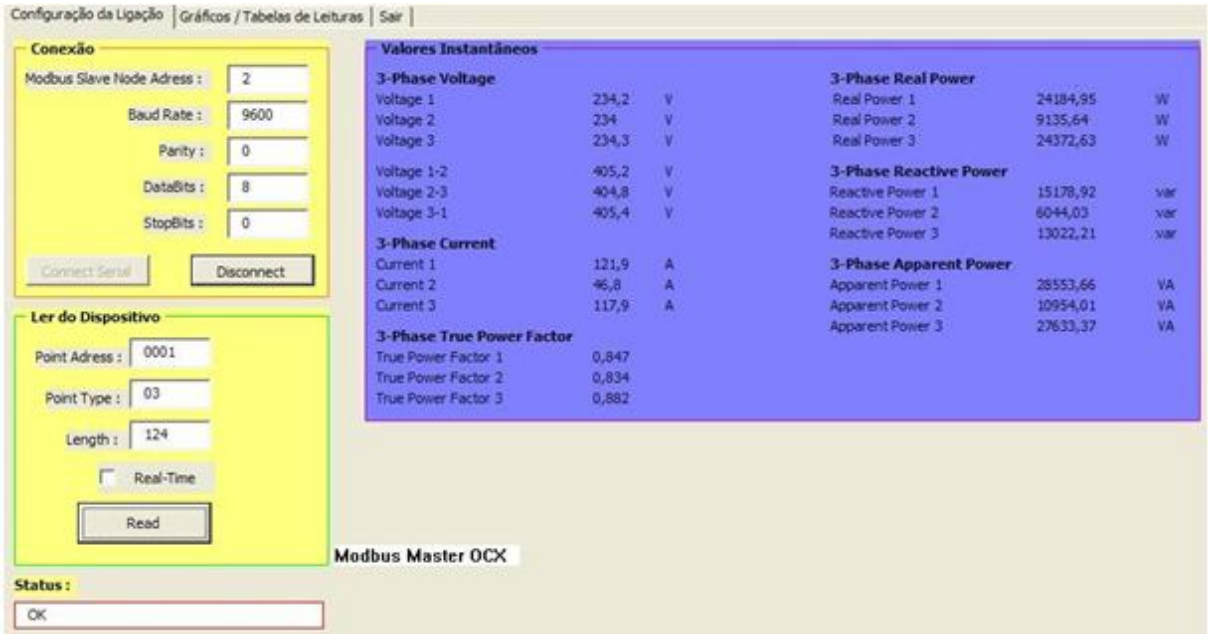


Figura 3.27 - Aplicação para análise de consumo em *Real-Time*

A segunda variante da aplicação – *LOGGER* -, Figura 3.28, tem como objectivo a recolha dos dados relativos às potências activa e reactiva de todos os dispositivos considerando intervalos de 24 horas. Os dispositivos podem estar ligados através da Internet ou de ligações de área local. No primeiro caso, um único servidor é capaz de recolher toda a informação de vários analisadores ligados a um conjunto de conversores. No segundo caso, cada rede local, isto é, se existir mais do que uma, necessita de um servidor.

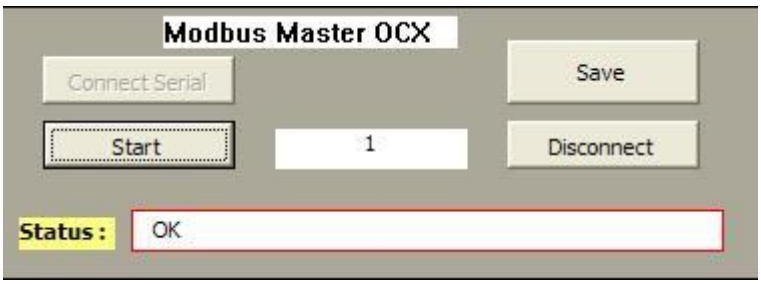


Figura 3.28 - Aplicação para recolha de dados

A aplicação *LOGGER* permite a recolha sistemática dos dados necessários para a gestão de energia bem como criação de relatórios, possibilitando a consulta remota ou envio automático por e-mail.

Esta aplicação não substitui a aplicação *ONLINE*. Permite a recolha dos dados e produz os relatórios necessários à gestão de energia. Ajusta-se aos quatro períodos de facturação anuais, cada um programado de forma dinâmica de maneira a que os períodos diários (horas do dia) estejam de acordo com o período presentemente a decorrer.

O *LOGGER* aumenta a transparência relativamente aos gastos energéticos pois acrescenta detalhe relativamente ao consumo energético decorrente de cada actividade. Assim, ganha-se informação essencial para ajustar os tarifários contratados e a política de investimento em novos equipamentos de energia. Com esta informação é possível procurar processos e procedimentos de operação mais eficazes, que conduzam a um desempenho superior, comparando os consumos entre instalações e equipamentos análogos.

Através dos relatórios de consumos de energia de cada rede implementada e consequentemente de cada analisador nela instalada, é possível facturar os valores de energia separadamente.

A informação recolhida é armazenada em ficheiros distintos. Isto faz sentido. Ou seja, se por algum motivo se pretende comparar os consumos de dois departamentos (por exemplo), é fácil saber qual o dispositivo correspondente a cada um e quais os dados a analisar.

Capítulo 4 - Implementação

Qualquer arquitectura de rede nasce de uma ideia conceptual. Após efectuar um esboço da rede a implementar, há que escolher os componentes da rede de comunicação, tendo em conta o protocolo seleccionado. Implementada a solução proposta, recolhem-se dados de consumo energético, analisam-se e apresentam-se os resultados.

4.1 Arquitectura de rede do sistema – Ideia conceptual

Gerir a energia é um processo contínuo. Os sistemas eléctricos em grandes complexos mudam constantemente, quer seja devido a modificações no sistema já existente quer pela inclusão de novos equipamentos.

Para gerir, deve-se primeiro medir e compreender o sistema em causa. Só depois se devem determinar as soluções e proceder à sua implementação, Figura 4.1.

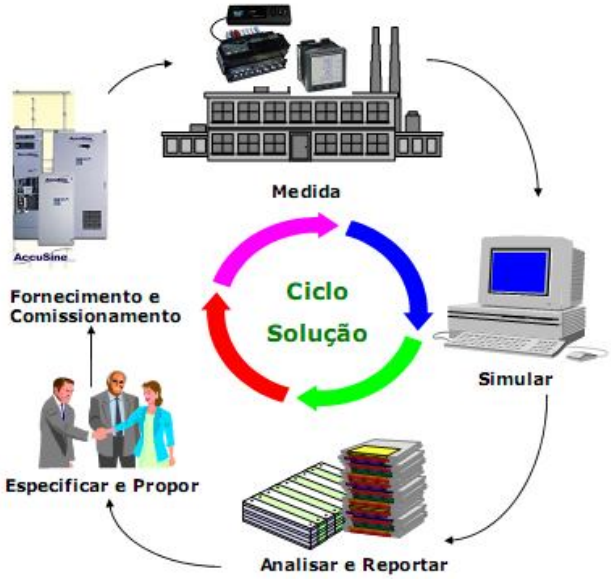


Figura 4.1 – Ciclo de solução [33]

O Energy Monitor apresenta-se como a solução para este tipo de situações. Esta aplicação permite-nos não só reunir a informação de diferentes pontos de medida, quer sejam eles no mesmo edifício ou não, mas também analisá-la de forma precisa, Figura 4.2. Após essa análise, torna-se fácil especificar e propor medidas para uma melhor gestão energética.

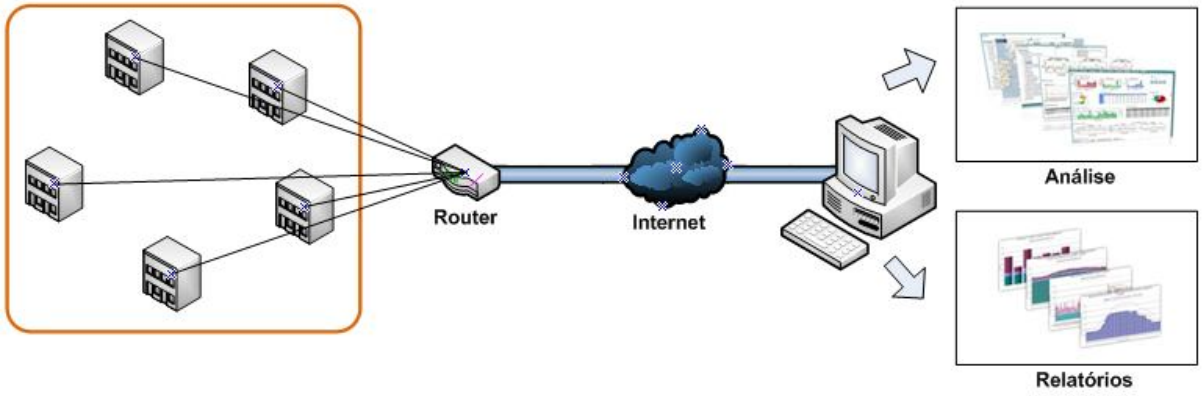


Figura 4.2 – Ideia conceptual do Energy Monitor

Como exemplo, este sistema pode ser utilizado num pequeno edifício ou numa sala técnica isolada. Um ou mais dispositivos de monitorização/medida podem ser ligados a um computador via modem ou mesmo através da Ethernet, Figura 4.3.

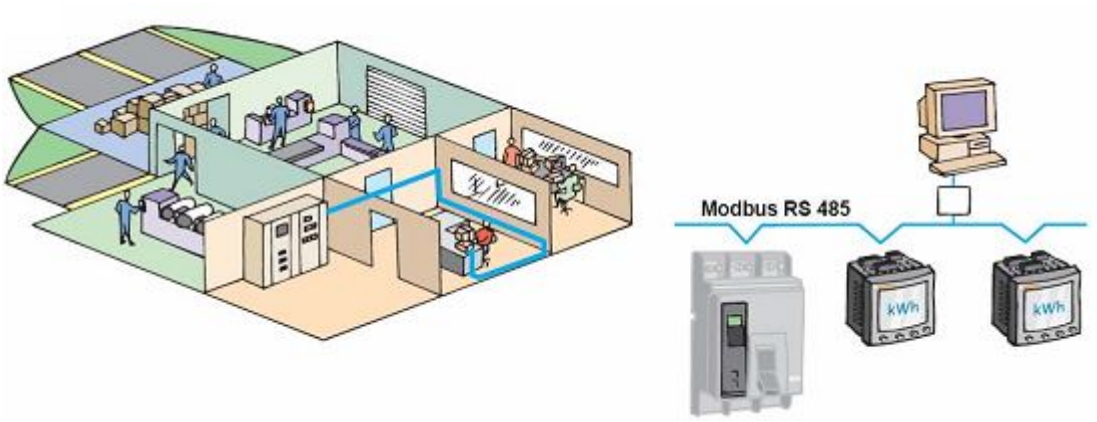


Figura 4.3 – Instalação do Energy Monitor num pequeno edifício [33]

Caso os postos de medida estejam localizados em edifícios diferentes, o Energy Monitor apresenta-se também como solução para a monitorização dos consumos de energia, Figura 4.4.

Neste caso, os dispositivos de medida são ligados a modems ou routers que servem posteriormente para integração na rede Ethernet da companhia. Desta forma, todos os sectores da companhia ligados à Ethernet ou à internet poderão ter acesso directo aos dados de uma determinada instalação eléctrica.

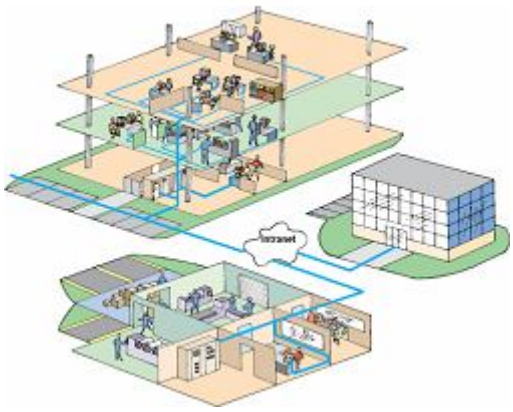


Figura 4.4 – Aplicação em edifícios separados [33]

A aplicação do Energy Monitor em edifícios acarreta inúmeras vantagens. Com esta solução, é possível obter custos de energia precisos dos locatários. Assim, pode-se gerir a energia como um custo variável para reduzir os custos de produção e aumentar o resultado líquido da exploração do edifico. Por outro lado, possibilita a eficiência energética como uma vantagem competitiva para atingir os rácios dos edifícios verdes bem como o suporte a objectivos sociais/ambientais da companhia.

Já a nível industrial, Figura 4.5, pode-se ter uma instalação numa fábrica com número variável de utilizadores interligados pela intranet. As unidades de medida são ligadas e integradas na rede Ethernet já existente na empresa conduzindo a uma gestão partilhada da instalação eléctrica por parte de diferentes departamentos.

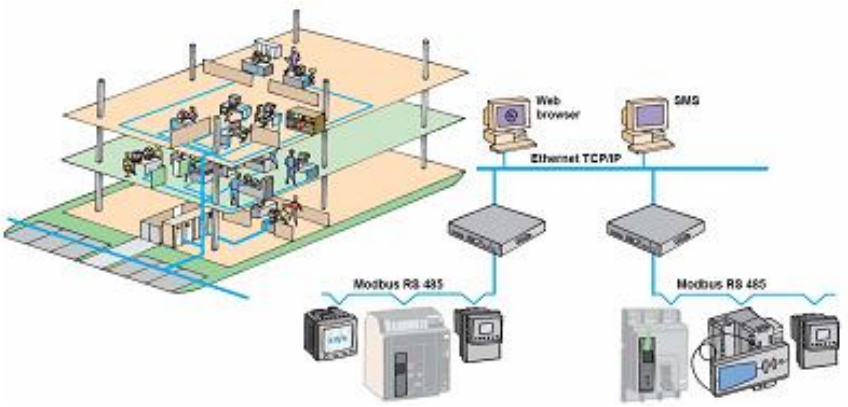


Figura 4.5 – Solução a nível industrial [33]

As vantagens relativamente à implementação desta solução na indústria são evidentes:

Eficiente geração ou compra de energia relacionada com o custo real:

- Monitorizar e controlar os equipamentos de geração;
- Analisar as opções de compra de energia;
- Escolher gerar, comprar ou reduzir inteligentemente.

Produção eficiente:

- Custos mais baixos de energia;
- Qualidade de Energia satisfatória para manter o processo.

Conservação e Eficiência Energética:

- Melhor conhecimento e contabilização do uso da energia para utilizadores finais.

Tudo isto permitirá, ao responsável pela empresa, negociar os contratos de energia e gerir a geração/consumos e o funcionamento da instalação.

Vistas as vantagens que o Energy Monitor acarreta para a indústria bem como a ideia conceptual relativamente ao seu funcionamento, é chegado o momento de compreender a sua topologia de rede.

4.2 Rede de comunicação e seus componentes

Na Figura 4.2, apresenta-se uma ideia geral da solução proposta. O que ressalta à vista é a possibilidade de existirem diferentes pontos de medida espalhados num campus - por exemplo -, um concentrador ao qual estão ligados os dispositivos de medida e um sistema supervisor que fará a análise dos dados.

Pois bem, estudando pormenorizadamente a topologia de rede já apresentada, há que identificar os componentes a utilizar e, as ligações entre eles.

Na Figura 4.6 é apresentada a topologia de rede do Energy Monitor. Os componentes escolhidos para a sua construção serão apresentados posteriormente.

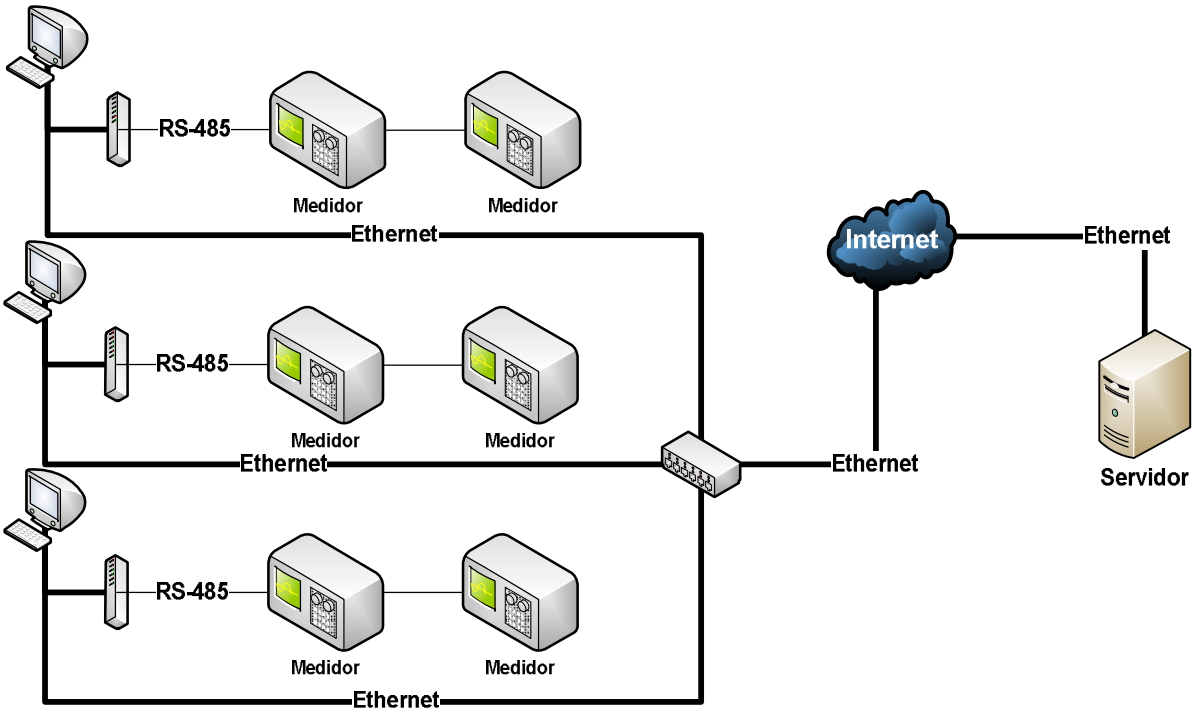


Figura 4.6 – Topologia de Rede

Tendo a arquitectura de rede montada, é altura de escolher os dispositivos que a compõem. De notar que estes são escolhidos com base no protocolo que se decidiu utilizar para implementar este trabalho.

De acordo com a Figura 4.6, para construir a rede de comunicação, serão precisos medidores, concentradores, computadores e conversores.

Os medidores escolhidos são os UPT210 Universal Power Transducer, Figura 4.7. Trata-se de um contador de energia multifunção capaz de medir os parâmetros eléctricos de um sistema trifásico [34].

Figura 4.7 – Medidor de energia UPT210 [34]

Fornece medições precisas mesmo quando em presença de ondas distorcidas. Os parâmetros de funcionamento deste dispositivo são facilmente programáveis através do seu

painel de controlo. Um LCD na parte frontal permite a visualização dos parâmetros que se pretende medir. A sua ligação é feita a outros dispositivos utilizando a interface RS-485.

Este dispositivo tem um custo relativamente baixo quando comparado com outros analisadores de energia. Pode operar sozinho ou como parte integrante de uma extensa rede de monitorização e gestão de energia [35].

O UPT210 pode ainda substituir vários dispositivos analógicos de uma só função, como é o caso dos voltímetros, amperímetros, wattímetros, analisadores de factor de potência, contadores de energia, etc. Possibilita ainda uma rápida integração nos sistemas a analisar pois é dotado de ligações a pinças flexíveis do tipo Rogowski[3].

Como já foi referido, o UPT210 tem apenas como interface de comunicação o padrão RS-485. Assim sendo, torna-se necessário a existência de um módulo que possibilite a conversão de RS-485 para Ethernet para que o sistema supervisor consiga comunicar com os medidores de energia. Para efectuar a referida conversão, optou-se pelo PD8(RS-485/Ethernet Converter), Figura 4.8.

Figura 4.8 – PD8 (Conversor RS-485/Ethernet) [36]

Este conversor pode funcionar de duas maneiras distintas: através de *Real-Port* ou usando o protocolo UDP com a *serial bridge* [36].

Neste caso em particular, o conversor actua através do serviço de *Real-Port*, uma vez que existe a necessidade de utilizar um computador de forma a armazenar a informação proveniente dos medidores UPT210. Usando o conversor no modo *Real-Port*, apenas um computador pode ser ligado ao conversor em simultâneo. No caso em estudo, para usar um

[3] A utilização das pinças Rogowski transforma a montagem de transformadores de intensidade em barramentos em carga, num trabalho simples e rápido, podendo poupar-se muitos minutos de trabalho. A robustez eléctrica destas pinças torna-as indicadas para qualquer trabalho pois podem funcionar desde mA até centenas de kA. Outra grande vantagem é o seu peso - são leves o suficiente para que possam ser montadas sobre o próprio condutor sem causar qualquer dano.

conversor PD8 com o protocolo Modbus, é necessário utilizar um computador para controlar os intervalos de tempo entre as transmissões de dados.

Apesar de não estar representado na Figura 4.6, são utilizados transformadores de corrente, como o mostrado na Figura 4.9.

O transformador de corrente é usado quando se pretende medir correntes eléctricas. Quando a corrente num circuito é demasiadamente elevada a ponto de não se conseguir ligar directamente dispositivos de medida, recorre-se à utilização de transformadores de corrente. Estes elementos produzem uma corrente mais baixa, directamente proporcional à corrente no circuito (neste caso, 50:5 CT), podendo-se assim ligar directamente a dispositivos de medição ou análise.

Figura 4.9 – Transformador de corrente

Um dos aspectos que ressalta à vista na topologia de rede apresentada na Figura 4.6 é o uso constante da Ethernet para interligar os dispositivos. Nomeadamente, do concentrador até ao servidor ou sistema supervisor.

A Ethernet IEEE 802.3 é um protocolo de rede de longa data que ganhou aceitação universal em todo o mundo. Trata-se de um padrão aberto, suportado por muitos fabricantes e sua infra-estrutura está amplamente disponível e largamente instalada. Por conseguinte, a sua gama de protocolos TCP / IP é utilizada no mundo inteiro e, até mesmo, serve como base para o acesso à World Wide Web. Como muitos dispositivos já suportam Ethernet, é natural expandi-la até às aplicações industriais.

Assim como acontece com a Ethernet, o Modbus é livremente disponível, acessível a qualquer um e, amplamente suportado por muitos fabricantes de equipamentos industriais. É também de fácil compreensão e um candidato natural para uso na criação de novos standards de comunicação industrial.

Com tanta coisa em comum, o casamento do protocolo de aplicação Modbus com o tradicional IEEE 802.3 Ethernet constitui um poderoso padrão de comunicação industrial - Modbus TCP / IP. E porque o Modbus TCP / IP partilha a mesma camada física e de enlace de

dados que o IEEE 802.3 Ethernet e usa o mesmo conjunto de protocolos TCP / IP, continua totalmente compatível com a já instalada infra-estrutura Ethernet de cabos, conectores, interface de placas de rede, *hubs* e *switches*.

Isto é importantíssimo pois, para grandes complexos, não haverá necessidade de inserir uma nova cablagem para implementar este tipo de solução ou mesmo para inserir novos componentes que utilizem o protocolo Modbus. Basta utilizar a rede de cabos existente, o que reduz de forma drástica os custos de implementação. Este é mais um dos aspectos que vêm reforçar a escolha do Modbus para este trabalho.

4.3 Instalação

Nesta secção far-se-á uma análise relativamente à rede implementada e à sua topologia. Pretende-se dar a conhecer o RS-485, o papel do hardware apresentado na secção anterior e, finalmente, apresentar a rede montada.

4.3.1 RS-485 – Especificação e Utilização

Tendo em conta que uma aplicação consiste em vários dispositivos em lugares diferentes, certamente, será necessário um meio de comunicação entre eles.

Apesar de o RS-232 (***R**ecommended **S**tandard*) ser a interface mais utilizada na comunicação série, possui algumas limitações. O padrão RS-485, criado em 1983 pela *Electronics Industry Association* (EIA), permite uma comunicação multiponto bastante robusta que, tem vindo a ser muito utilizada na indústria em controlo de sistemas e em transferências de dados para pequenas quantidades e taxas de transmissão até 10Mbps [37].

No RS-232, os sinais são representados por níveis de tensão referentes à terra. Há um fio para transmissão, outro para recepção e o fio terra para referência dos níveis de tensão. Este tipo de interface é útil em comunicações ponto-a-ponto a baixas velocidades de transmissão. Devido à necessidade de um fio de terra comum entre os dispositivos, há limitações do comprimento do cabo a apenas algumas dezenas de metros. Os principais problemas são a interferência e a resistência do cabo.

O padrão RS-485 utiliza apenas um par de fios. Pode funcionar em modo *Half-duplex* ou em *Full-duplex* permitindo uma comunicação de forma diferencial e até 32 terminais remotos de comunicação por nós da rede – recorrendo à introdução de repetidores, o número de nós pode ser aumentado para centenas ou mesmo milhares, Figura 4.10 [38].

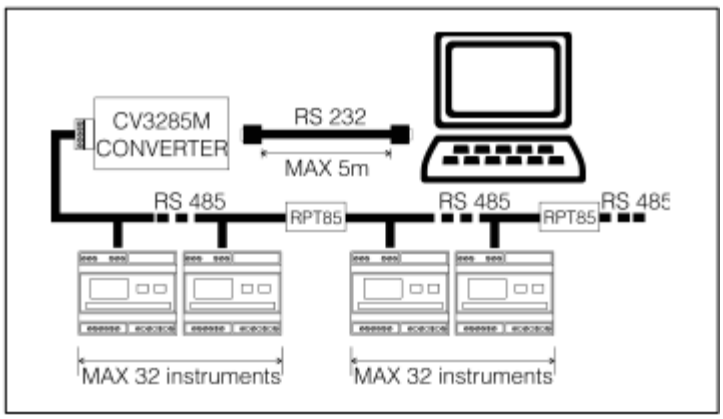


Figura 4.10 – Interface RS-485 com mais de 32 terminais remotos [38]

No que diz respeito a distâncias de transmissão, o RS-485 é bastante mais robusto que o RS-232. No primeiro caso, as distâncias chegam aos 1200m enquanto o RS-232 chega apenas aos 15 metros, Tabela 4.1.

Quanto maior a distância a ser percorrida pelos dados, menor será a taxa de transmissão. Assim, têm-se taxas de comunicação de até 10 Mbps quando consideramos distâncias curtas entre os terminais remotos (12m) e, de 100Kbps para distâncias de 1200m.

Tabela 4.1 - Características do RS232, RS-423, RS-422 e RS-485 [38]

	RS232	RS423	RS422	RS485
Differential	no	no	yes	yes
Max number of drivers Max number of receivers	1 1	1 10	1 10	32 32
Modes of operation	half duplex full duplex	half duplex	half duplex	half duplex
Network topology	point-to-point	multidrop	multidrop	multipoint
Max distance (acc. standard)	15 m	1200 m	1200 m	1200 m
Max speed at 12 m Max speed at 1200 m	20 kbs (1 kbs)	100 kbs 1 kbs	10 Mbs 100 kbs	35 Mbs 100 kbs
Max slew rate	30 V/µs	adjustable	n/a	n/a
Receiver input resistance	3..7 kΩ	≧ 4 kΩ	≧ 4 kΩ	≧ 12 kΩ
Driver load impedance	3..7 kΩ	≧ 450 Ω	100 Ω	54 Ω
Receiver input sensitivity	±3 V	±200 mV	±200 mV	±200 mV
Receiver input range	±15 V	±12 V	±10 V	-7..12 V
Max driver output voltage	±25 V	±6 V	±6 V	-7..12 V
Min driver output voltage (with load)	±5 V	±3.6 V	±2.0 V	±1.5 V

Tipicamente, o RS-485 é utilizado com um único computador ligado a diversos dispositivos endereçáveis que compartilham o mesmo cabo – o endereçamento é tratado pela unidade remota.

Este padrão apenas especifica as características eléctricas e os modos de operação da rede, não especificando nem recomendando o protocolo a utilizar.

As principais vantagens do padrão RS-485 resumem-se a [39]:

- Redes locais baratas, quando comparadas a outras (*FieldBus, Ethernet*, por exemplo);
- Flexibilidade de configuração;
- O utilizador define, projecta e testa o seu próprio protocolo de comunicação;

Tendo em linha de conta as vantagens atrás apresentadas, o RS-485 foi o escolhido para este trabalho em particular, uma vez que se não é especificado o protocolo de comunicação, podem-se usar protocolos abertos (como é o caso do protocolo Modbus), bem definidos e testados, sem que haja necessidade de pagar a fabricantes.

4.3.2 Detalhes da Rede Implementada

A rede implementada encontra-se dividida em duas áreas distintas: a implementação de um Mestre e a implementação dos Escravos, Figura 4.11.

O Mestre, é composto por um computador onde está implementado o software de monitorização, cujo papel é arbitrar os tempos de intervalo entre a transmissão e a recepção de mensagens enviadas aos dispositivos de medida, recolher a informação e armazená-la.

A implementação dos Escravos compreende toda a aparelhagem envolvente, nomeadamente, os dispositivos de medida (UPT210), os transformadores de corrente (CT) e os conversores PD8 que, permitem a comunicação entre a rede local ou internet e os dispositivos equipados com a interface RS-485.

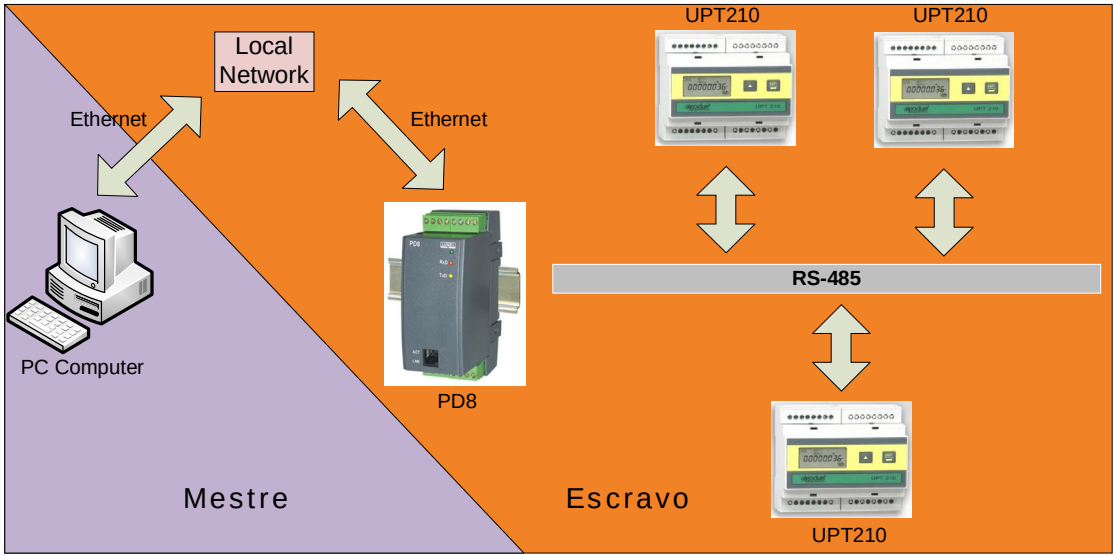


Figura 4.11 – Rede implementada

Seguidamente, apresentam-se os pormenores relativamente à implementação do Mestre e dos Escravos.

4.3.3 O Mestre

O conversor PD8 é ligado ao computador Mestre através do serviço Real-Port. Este serviço possibilita a comunicação, através de Ethernet com qualquer computador que opere com o sistema operativo Windows.

O serviço *Real-Port* leva à adição de uma porta COM virtual à lista corrente de portas COM disponíveis, Figura 4.12. Esta nova porta não terá um suporte físico uma vez que é uma porta virtual. Assim, pode-se substituir a comunicação série pela Ethernet.

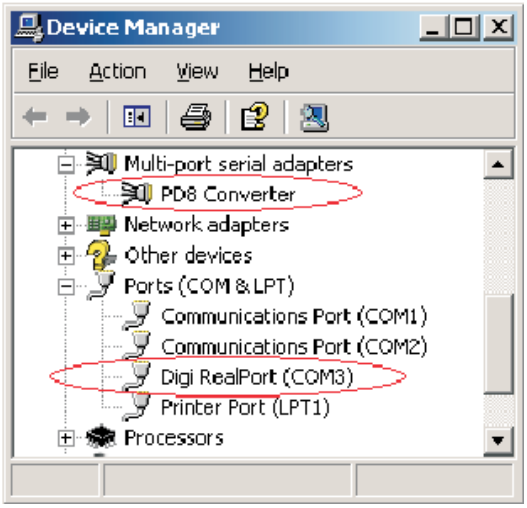


Figura 4.12 – Lista de portas COM no sistema Windows

Devido às especificações de fabrico do conversor PD8, apenas pode existir uma conexão ao serviço *Real-Port* em simultâneo. Isto significa que apenas um computador, com qualquer endereço IP pode comunicar com o computador através da porta série virtual. Para outro computador, este serviço está inacessível no momento e, será lançada uma mensagem de erro se ele tentar aceder a esse serviço.

Criada a porta série virtual, torna-se necessário proceder à sua configuração. Podem-se alterar os valores relativos ao *baud rate*, bits de dados, paridade, número de stop bits e controlo de fluxo, Figura 4.13. Dever-se-á ainda ajustar os parâmetros relativos ao tempo de resposta dos escravos – tendo em conta o protocolo em uso (neste caso o Modbus) e a interface RS-485 – e à capacidade da rede Ethernet à qual estará ligado o PD8.

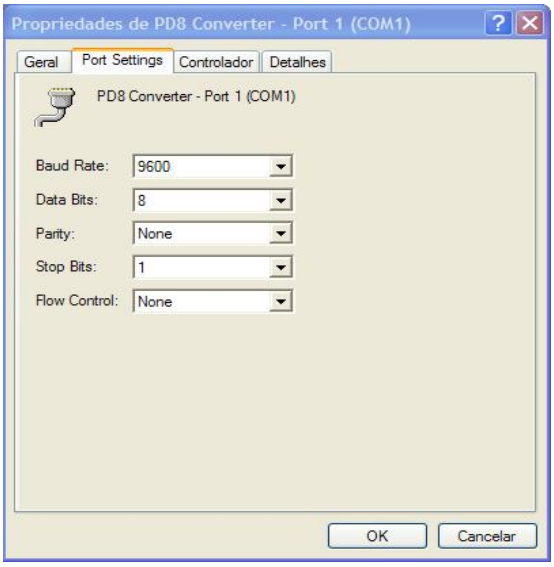


Figura 4.13 – Propriedades da porta COM virtual

Configurada a porta série COM virtual, liga-se o PD8 ao computador. Tal como outros dispositivos de rede, o conversor PD8 requer a configuração de alguns parâmetros, nomeadamente no que se refere ao protocolo IP.

Assim, os parâmetros a configurar são: endereço IP do PD8, máscara da sub-rede e *gateway*. Estes dados são obtidos do computador ao qual o conversor está conectado. De modo a facilitar este processo, utiliza-se uma aplicação fornecida pelo fabricante – PD8Config.exe, Figura 4.14.

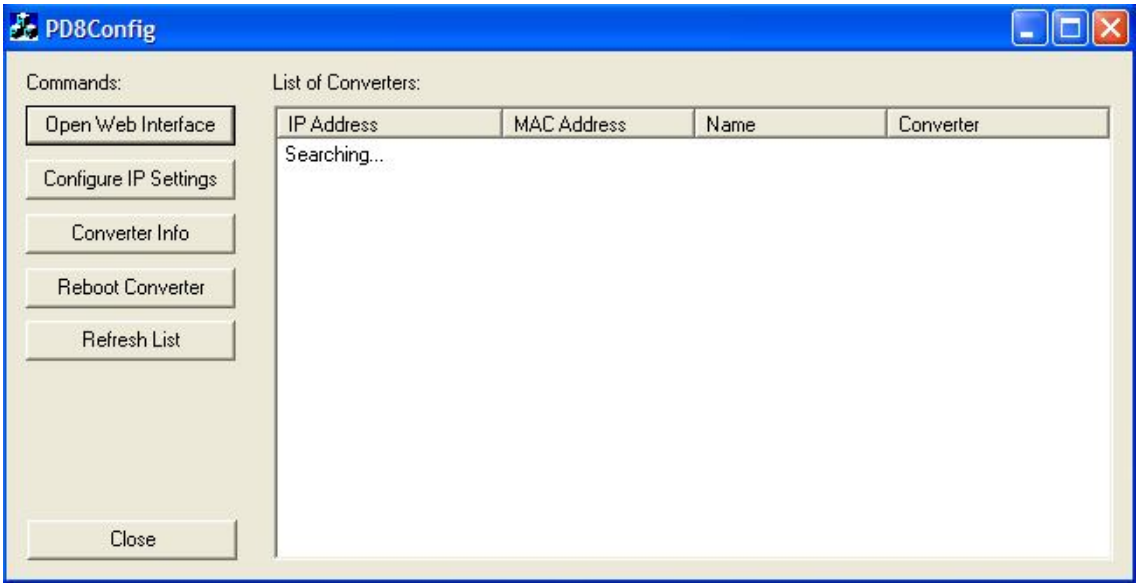


Figura 4.14 – Interface da aplicação PD8Config

Com o conversor ligado à rede local, executa-se o PD8Config. Esta aplicação faz um varrimento à rede, listando todos os conversores a ela conectados, indicando o endereço IP, o endereço MAC e o nome do mesmo, Figura 4.15. Caso o endereço de IP apareça na forma 0.0.0.0., significa que na rede em que questão, o serviço DHCP não está activo e, dessa forma, os parâmetros do PD8 têm de ser colocados manualmente.

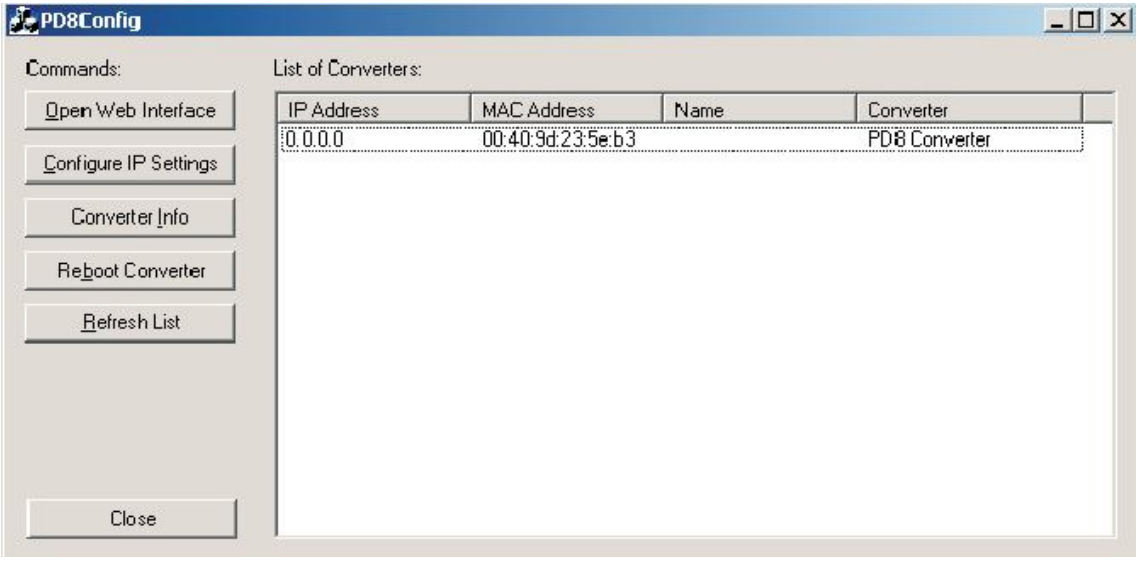


Figura 4.15 – Interface do PD8Config após o varrimento da rede

Neste caso, para introduzir os parâmetros correctos, manualmente, basta seleccionar o dispositivo que se pretende e partir para a sua configuração, como exemplificado na Figura 4.16.

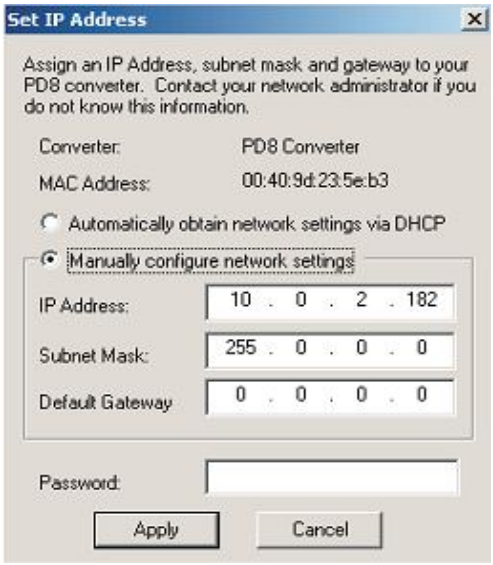


Figura 4.16 – Alteração dos parâmetros do PD8

Tendo configurado todos os conversores PD8 de forma correcta, o Mestre fica habilitado a identificar qualquer conversor na rede local de forma inequívoca e, consequentemente, poderá endereçar qualquer UPT210 (Escravos) a ela conectado.

Tendo implementado o Mestre na rede apresentada, é agora altura de implementar os Escravos.

4.3.4 Os Escravos

Esta parte da rede compreende os dispositivos de medida (UPT210) que, ligados aos aparelhos pretendidos e, posteriormente conectados à rede, permitem efectuar leituras de diversas grandezas, Figura 4.17.

INSTANTANEOUS MEASUREMENTS		
PHASE VOLTAGE *	V_1 - V_2 - V_3 [V]	●
LINE VOLTAGE	V_{12} - V_{23} - V_{31} [V]	●
SYSTEM VOLTAGE	V [V]	●
LINE CURRENT *	I_1 - I_2 - I_3 - I_N [A]	■
SYSTEM CURRENT	I [A]	■
POWER FACTOR *	PF_1 - PF_2 - PF_3	●
SYSTEM POWER FACTOR	PF	●
APPARENT POWER *	S_1 - S_2 - S_3 [VA]	■
SYSTEM APPARENT POWER	S [VA]	■
ACTIVE POWER *	P_1 - P_2 - P_3 [W]	■
SYSTEM ACTIVE POWER	P [W]	■
REACTIVE POWER *	Q_1 - Q_2 - Q_3 [var]	■
SYSTEM REACTIVE POWER	Q [var]	■
FREQUENCY	f [Hz]	●
PHASE REVERSAL	123 / 132	●
STORED DATA		
SYSTEM ACTIVE ENERGY	[Wh]	■
SYSTEM APPARENT ENERGY	[VAh]	■
SYSTEM LAGGING REACTIVE ENERGY	[varh ind]	■
SYSTEM LEADING REACTIVE ENERGY	[varh cap]	■
● = Standard ■ = Bi-directional value (BIDIR option only)		

Figura 4.17 – Grandezas suportadas pelo UPT210 [40]

Tendo em conta os objectivos do sistema a desenvolver, as grandezas mais importantes que o analisador de energia nos pode dar são: tensão, corrente e o factor de potência. Segundo as informações fornecidas pelo fabricante do UPT210, os valores relativos às potências, tanto activa como reactiva bem como o valor da energia não são medidos directamente.

Os valores das potências e da energia são obtidos através de cálculos baseados nas tensões, correntes e factor de potência que, são obtidos directamente.

Assim sendo, a parte da rede a tratar neste momento é a apresentada na Figura 4.18.

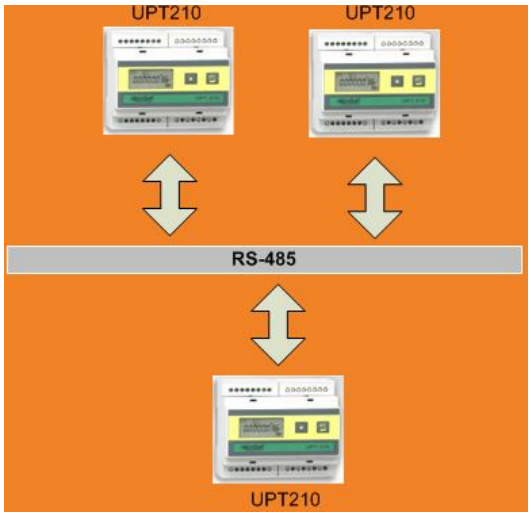


Figura 4.18 – Ligação dos UPT210 à interface RS-485

Consoante o sistema eléctrico em causa (monofásico ou trifásico), o esquema de ligações dos UPT210 altera-se. Assim sendo, há que ter em linha de conta o sistema em que se está a operar bem como as possíveis intensidades de corrente que daí advêm. De modo a evitar correntes muito altas, que não pudessem ser suportadas pelos analisadores, é necessário introduzir alguns transformadores de corrente entre os dispositivos de medida e os equipamentos onde se pretende efectuar as leituras, Figura 4.19.

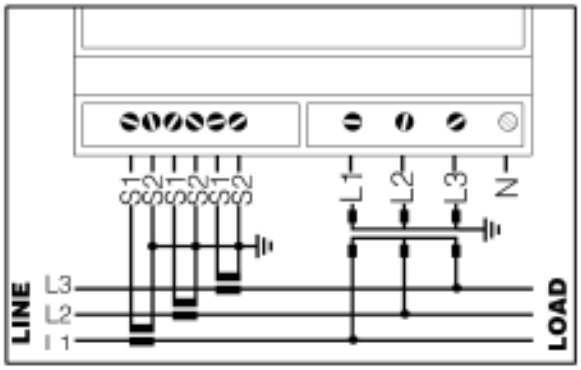


Figura 4.19 – Exemplo de ligação do UPT a um sistema trifásico usando 3 transformadores de corrente [40]

Uma vez que o RS-485 possibilita a comunicação multiponto, há que ter em atenção alguns aspectos a evitar, nomeadamente articulações e shunts. Por outras palavras, é

necessário ligar os fios ao primeiro contador, deste ao segundo e por aí em diante até ao final da linha, como mostra a Figura 4.20.

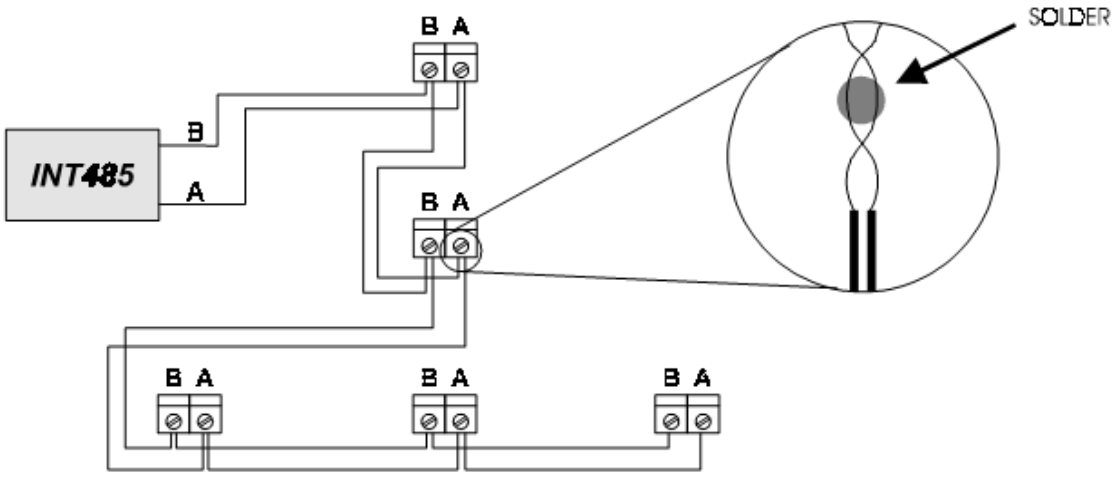


Figura 4.20 – Ligação correcta numa interface RS-485 [39]

Após a ligação de todos os dispositivos de medida, procede-se à sua configuração, nomeadamente no que se refere ao seu endereço. Isto é bastante importante pois cada contador terá o seu endereço, Figura 4.21. Só assim o Mestre poderá comunicar com o dispositivo pretendido e, recolher as leituras necessárias. Os endereços lógicos, tendo em conta o protocolo Modbus vão desde $01-$F7 (1-247).

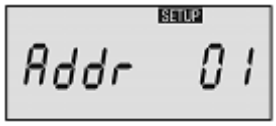


Figura 4.21 – Display de configuração do endereço do UPT210

Não é apenas o endereço do dispositivo que se deve configurar. Outros aspectos, não menos importantes devem ser tidos em atenção:

- *Baud Rate*

A velocidade da comunicação deve ser concordante com a especificada para o PD8. Pode tomar os seguintes valores: 2400, 4800, 9600, 19200, 38400, 57600.

- *Protocolo*

Existem dois possíveis protocolos a seleccionar: Standard ASCII e o Modbus. Neste caso, em particular, usa-se o Modbus e, deve-se ainda escolher qual o modo de transmissão, ou seja, ASCII ou RTU.

- *Norma de transformação*

Ao usar transformadores de corrente, a norma de transformação deve ser colocada no medidor correspondente de modo a que as leituras efectuadas sejam as correctas.

Nesta fase, os UPT210 estão correctamente configurados e ligados à rede. O mestre, como já se viu, está também configurado e ligado ao conversor PD8 possibilitando a comunicação com os escravos. Assim sendo, tem-se uma rede completamente configurada e pronta a funcionar.

De agora em diante, é possível enviar mensagens aos Escravos solicitando informações de leitura ou apenas que estes efectuem algum procedimento.

4.4 Aquisição de Dados

Como já foi referido anteriormente, a comunicação entre o Mestre e os Escravos é feita através do protocolo Modbus. Existindo a necessidade de armazenar e tratar os dados, nada melhor que utilizar uma folha de cálculo digital para o efeito – neste caso, utilizou-se o Excel.

Para além de ser uma ferramenta presente quase em todos os computadores, o Excel, permite importar, organizar e explorar grandes conjuntos de dados, criar gráficos totalmente estruturados, elaborar tabelas, "observar" tendências importantes e encontrar excepções nos dados.

A todas estas propriedades, pode-se juntar o facto de o Microsoft Excel poder trabalhar com controladores já desenvolvidos, bem como com inúmeras *toolboxes* disponibilizadas livremente pela Microsoft.

Posto isto, resolveu-se utilizar o controlador "Modbus Master" para auxiliar a comunicação do Mestre com os Escravos.

4.4.1 Controlador Modbus Master

O Modbus Master Control (mbMasterV7) é um controlador distribuído pela WinTECH Software Design que pode ser interligado com a componente do Visual Basic do Excel.

Este controlador permite que o Visual Basic aceda de forma simples e rápida a determinados pontos de informação contidos num escravo Modbus ligado a um PC. O dispositivo Modbus pode ser ligado directamente à porta COM do computador ou, pode ser acedido através de uma rede que utilize o protocolo de comunicação standard Modbus/TCP. Existe ainda a possibilidade de comunicação através de modem, utilizando a biblioteca Win32 TAPI (Telephony APlication Interface).

A aplicação, usando este controlador, poderá estabelecer múltiplas ligações a diferentes portas COM e/ou a dispositivos de rede. Cada ligação funcionará de forma independente, permitindo assim a comunicação assíncrona a múltiplos dispositivos espalhados por diversas redes Modbus.

O mbMasterV7 permite ao utilizador definir as propriedades físicas da ligação (*baud rate*, paridade, número de *stop bits*, etc.) e os métodos de conexão à porta COM, à rede ou a um dispositivo TAPI. Cada ligação é identificada univocamente por um sinal retornado à aplicação através deste controlador – quando a ligação é estabelecida [41]. Seguidamente a aplicação inicia o envio de mensagens para o dispositivo que pretende contactar, usando o comando *PollModbus* e/ou *WriteModbus.* Todas as mensagens são processadas em "*backgroung*" permitindo desta forma que a aplicação lógica continue a funcionar enquanto as mensagens são formatadas e enviadas aos respectivos dispositivos.

O controlador é ainda responsável pelo processamento de todas as mensagens de resposta, verificando se foram correctamente recebidas e lançando o evento de resposta – *SlaveResponse* – para a aplicação, informando os resultados da transmissão. Depois de processado o evento de *SlaveResponse*, a aplicação poderá obter os resultados das mensagens de escrita ou leitura anteriormente enviadas.

O processo anteriormente referido pode ser observado na Figura 4.22.

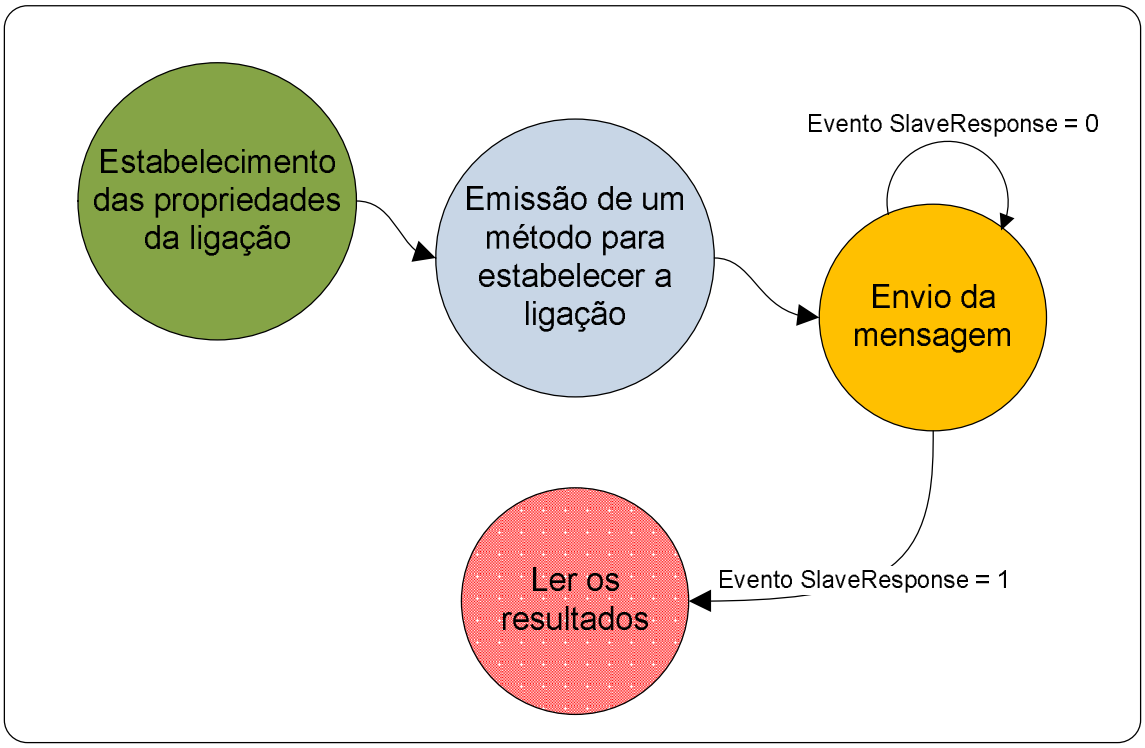


Figura 4.22 – Ciclo de preparação, envio, recepção e leitura de uma mensagem Modbus

4.4.2 Descrição do Controlador – Modbus Master

O controlador mbMasterV7 consiste num conjunto de propriedades e métodos para definir as características operacionais de uma ligação física a uma rede Modbus. Assim, a aplicação pode iniciar o envio de mensagens e, consecutivamente, a leitura das respostas volvidas dos dispositivos escravos ligados à rede.

O controlador suporta os seguintes tipos de mensagens Modbus (Tabela 4.2):

Tabela 4.2 - Mensagens suportadas pelo controlador Modbus Master

Código da Função	Nome
01	Read Coil Status
02	Read Input Status
03	Read Holding Registers
04	Read Input Registers
05	Write Single Coil
06	Write Single Register
15	Write Multiple Coils
16	Write Multiple Registers

As funções anteriormente apresentadas já foram descritas no Capítulo 3, pelo que não serão descritas novamente nesta secção.

Como referido previamente, o controlador mbMasterV7 possui ainda um conjunto de propriedades que são apresentadas na Tabela 4.3. Estas propriedades permitem definir as características da ligação.

Tabela 4.3 - Propriedades do controlador Modbus Master (mbMasterV7)

Propriedade	Descrição
TrasmissionMode	0 = ASCII, 1 = RTU
TimeOut	Define o timeout associado ao protocolo, para este controlador. Ou seja, define o tempo de espera que a aplicação deverá aguardar pela resposta do escravo (após ter sido efectuado um pedido de informação). O tempo é especificado em milissegundos.
BaudeRate	300, 600, 1200, 2400, 4800, 9600, 14400, 19200
Parity	Define a paridade para as ligações à porta COM. 0 = Nenhuma, 1 = Ímpar, 2 = Par
StopBits	0 = 1 stopbit, 1 = 1.5 stopbits, 2 = 2 stopbits
DataBits	Número de bits de informação: 7, 8
TCPDevice	Define o servidor de Modbus/TCP para a ligação de rede. Pode ser na forma de endereço IP ou colocando o nome da máquina.
PhoneNumber	Define o número a marcar no caso de usar um modem (TAPI).

Apresentadas as propriedades, é altura de apresentar os métodos disponíveis neste controlador, Tabela 4.4. É através deles que se poderão enviar os pedidos de informação, ler as respostas dos escravos, verificar o estado das ligações, bem como ligar e desligar da rede.

Tabela 4.4 - Métodos disponíveis no controlador Modbus

Métodos	Descrição
HideControl()	Esconde o controlo
ConnectSerial()	Tenta a ligação através da porta COM.
ConnectModbusTCP()	Tenta a ligação a um servidor Modbus/TCP
DialTAPIDevice()	Tenta uma ligação remota, marcando o número do dispositivo TAPI pretendido.
PollModbus()	Inicia um pedido de leitura a um dispositivo Modbus (as mensagens só podem ser do tipo 01-04).
ReadResults()	Lê os resultados devolvidos após um PoolModbus().
FillWriteBuffer()	Prepara um buffer de informação para ser enviado a um dispositivo escravo.
WriteModbus()	Inicia um pedido de escrita (mensagens do tipo).
ReadResults()	Lê os resultados após um WriteModbus()
Disconnect()	Liberta a ligação bem como os recursos para que possam ser usados por outra aplicação do Windows.
HangupTAPI()	Permite desligar uma ligação, que esteja em progresso, a um dispositivo TAPI .
NumberOfTAPIDevices()	Devolve o número de dispositivos TAPI instalados na máquina em questão.
GetTAPIDeviceName()	Devolve o nome de cada dispositivo TAPI instalado na máquina.

Este controlador permite ainda o envio de mensagens configuradas pelo utilizador para os dispositivos escravos. Assim, os métodos apresentados na Tabela 4.5 permitem que a aplicação escreva uma *string* de bits, para ser enviada ao escravo e, consequentemente, consiga ler a resposta. De notar que estas mensagens não estão definidas e, será o utilizador a defini-las de acordo com as especificações do controlador.

Tabela 4.5 – Métodos de mensagens do utilizador

Métodos de mensagens do utilizador	Descrição
FillUserMsgBuffer()	Definição da string de bits a ser enviada ao escravo.
SendUserMsg()	Transmite a mensagem definida pelo utilizador ao dispositivo escravo.
ReadUserMsgResponse()	Lê o buffer de resposta.

As mensagens enviadas aos escravos e, a respectiva resposta podem ou não ser enviadas ou recebidas correctamente. Assim, se algum erro acontece, é disparado um evento que, depois de identificado revelará qual o erro ocorrido. Na Tabela 4.6 poderão observar-se os diferentes tipos de eventos disponíveis.

Tabela 4.6 – Eventos

Evento	Descrição
SlaveReadResponse	Este evento é disparado para alertar a chegada da resposta a um método PoolModbus().
SlaveWriteResponse	Disparado para significar a completa recepção de uma mensagem iniciada pelo método WriteModbus().
UserMsgResponse	Este evento é disparado, significando a conclusão de uma mensagem iniciada pelo método SendUserMsg().
ConnectionEstablished	Para a rede e ligações TAPI, este evento é disparado para significar que a ligação está concluída.
ConnectionDropped	Disparado sempre que a ligação de rede ou a um dispositivo TAPI falha. Este evento pode disparar aquando da tentativa da primeira ligação ou mesmo quando a ligação já está estabelecida, significando um problema com a mesma.

Apresentados os métodos, os eventos e os tipos de mensagens suportadas pelo utilizador, é hora de explicar mais em pormenor cada um deles.

4.4.3 Métodos, Mensagens e Eventos – Descrição

Cada método e mensagem recebe algum tipo de parâmetro de entrada e devolve parâmetros de saída. No caso dos eventos, algum parâmetro deverá ser devolvido à aplicação para que esta saiba o que ocorreu com a ordem ou processo que pretende que seja executado. Assim sendo, nesta secção, serão apresentadas as definições dos métodos, eventos e mensagens mais importantes. Dar-se-á ênfase aos parâmetros de entrada, de saída e ao tipo de funções que os implementam.

O método *ConnectSerial()*, Tabela 4.7, aceita um único parâmetro de entrada que identifica a porta COM a ser usada para a ligação. A porta designada é então aberta tendo em conta os valores definidos para o *BaudRate*, número de *DataBits*, número de *StopBits* e paridade. O valor devolvido por esta função será o *iHandle* - onde estarão os dados da ligação – e, deverá ser mantido pela aplicação que gere o controlador pois será passado posteriormente ao *PollModbus()*, *WriteModbus()*, *ReadResponse()*, etc.

Tabela 4.7 – Definição do método ConnectSerial()

Função	ConnectSerial
Parâmetros de entrada	Short Porta
Parâmetros de saída	Long iHandle
Exemplo	iHandle = mbMasterV71.ConnectSerial(1)

Se por algum motivo, a porta com não é aberta, ou seja, se ocorre uma falha neste capítulo, este método retornará um INVALID_HANDLE_VALUE com o valor -1.

No exemplo apresentado na Tabela 4.7, a aplicação tenta uma ligação através da porta COM 1.

O método *ConnectModbusTCP()*, Tabela 4.8, instrui o controlador a estabelecer uma ligação Modbus/TCP com o dispositivo de rede especificado pela propriedade *TCPDevice*, usando o número do porto especificado como argumento. No exemplo apresentado, o porto 502 é o porto standard utilizado pelo Modbus/TCP, tal como definido pela Modicom.

Tabela 4.8 – Método ConnectModbusTCP

Função	ConnectModbusTCP
Parâmetros de entrada	Short Porta
Parâmetros de saída	Long iHandle
Exemplo	iHandle = mbMasterV71.ConnectModbusTCP(502)

Este método devolverá um valor positivo (não nulo) caso a ligação tenha sido iniciada. No entanto, isto não confirma que a ligação tenha sido estabelecida. Assim, a aplicação deverá processar os eventos de *ConnectionEstablished* e *ConnectionDropped* de forma a verificar se a ligação está pronta a transaccionar mensagens.

DialTAPIDevice(), Tabela 4.9, é um método que instrui o controlador a abrir uma ligação remota TAPI a partir da marcação de um número de telefone especificado através da propriedade *PhoneNumber*. Enquanto a tentativa de ligação está a ser efectuada em *background*, o controlo é devolvido à aplicação.

Tabela 4.9 – Método DialTAPIDevice

Função	DialTAPIDevice
Parâmetros de entrada	Short DeviceIndex
Parâmetros de saída	Long iHandle
Exemplo	iHandle = mbMasterV71. DialTAPIDevice (0)

Ao usar este método, a aplicação deverá invocar em primeiro lugar o método *NumberOfTAPIDevices()* seguido do *GetTAPIDeviceName()* e só então executará o *DialTAPIDevice()*. Caso isto não aconteça correctamente, existirão problemas nas tentativas de ligações remotas.

Relativamente aos parâmetros de saída, o método agora apresentado retorna um valor (não nulo) caso a ligação tenha sido iniciada. No entanto e, em semelhança ao método *ConnectModbusTCP*, isto não quer dizer que a ligação tenha sido estabelecida. Desta forma, a aplicação deverá executar os eventos de *ConnectionEstablished* e *ConnectionDropped* de forma a verificar se a ligação está apta a trocar mensagens.

No exemplo apresentado atrás, pretende-se conectar usando o primeiro dispositivo TAPI definido.

O método *Disconnect()*, Tabela 4.10, fecha a ligação especificada libertando todos os recursos alocados para serem usados por outras aplicações do Windows. Caso a ligação tenha sido encerrada com sucesso o resultado deste método virá a TRUE.

Tabela 4.10 – Método Disconnect

Função	Disconnect
Parâmetros de entrada	Long ConnectionHandle
Parâmetros de saída	Boolean Status
Exemplo	Status = mbMasterV71. Disconnect (iHandle)

O exemplo mostra o encerramento da ligação estabelecida cujos parâmetros se encontram em *iHandle*.

Este método, *HangupCall*, Tabela 4.11, deve apenas ser usado para abortar uma tentativa de ligação TAPI, ou seja, quando a marcação está ainda em progresso. Após a ligação estar estabelecida, o seu fecho deve ser feito através do método *Disconnect()*.

Tabela 4.11 – Método HangupCall

Função	HangupCall
Parâmetros de entrada	Long ConnectionHandle
Parâmetros de saída	Void
Exemplo	mbMasterV71. HangupCall (iHandle)

No exemplo apresentado, o método aborta uma tentativa de ligação.

NumberOfTAPIDevices, Tabela 4.12, permite que a aplicação interrogue a biblioteca TAPI de forma a obter o número de dispositivos actualmente instalados na máquina. A aplicação apresentará uma lista com os respectivos dispositivos e seus nomes para que o utilizador possa seleccionar qual o que irá utilizar para efectuar a conexão.

Tabela 4.12 – Método NumberOfTAPIDevices

Função	NumberOfTAPIDevices
Parâmetros de entrada	Nenhum
Parâmetros de saída	Long NumberOfTAPIDevices
Exemplo	NumberOfDevices=mbMasterV71. NumberOfTAPIDevices()

A aplicação deverá começar por chamar os métodos *NumberOfTAPIDevices()* e *GetTAPIDeviceName()* antes de invocar o método *DialTAPIDevice()*. Se existirem falhas, a biblioteca de TAPI não será correctamente inicializada e, consequentemente, existirão erros aquando das conexões.

O exemplo anterior mostra a aplicação a determinar o número de dispositivos TAPI instalados.

GetTAPIDeviceName, Tabela 4.13 - permite que a aplicação interrogue a biblioteca de dispositivos TAPI para obter o nome de um determinado dispositivo instalado. Geralmente, como resultado da aplicação deste método, surge uma lista de dispositivos permitindo assim que o utilizador escolha qual o que pretende para efectuar a ligação.

Tabela 4.13 – Método GetTAPIDeviceName

Função	GetTAPIDeviceName
Parâmetros de entrada	String FirstModem
Parâmetros de saída	Long DeviceIndex
Exemplo	FirstModem =mbMasterV71. GetTAPIDeviceName (0)

A aplicação deverá invocar este método e o *NumberOfTAPIDevices()* antes de efectuar a marcação (*DialTAPIDevice()*). Caso existam falhas, a biblioteca TAPI não será correctamente inicializada – ocorrência de erros de conexão.

O exemplo apresentado mostra a aplicação a adquirir o nome do primeiro dispositivo TAPI.

O método *PollModbus()*, Tabela 4.14, inicia um pedido de leitura ao escravo indicado pelo *SlaveNodeAddress*. O *ConnectionHandle* define a ligação e, o seu valor deve ser igual ao devolvido pelo método *ConnectSerial()*, *DialTAPIDevice()* ou *ConnectModbusTCP()*.

Tabela 4.14 – Método PollModbus

Função	PollModbus
Parâmetros de entrada	Long ConnectionHandle, short SlaveNodeAddress, short ModbusFunctionCode, Long Address, short Length
Parâmetros de saída	Short Status
Exemplo	Status =mbMasterV71.PollModbus (iHandle,1,3,100,10)

O *SlaveNodeAdress*, tal como referido, indica o escravo Modbus que vai ser interrogado. O *ModbusFunctionCode* é o código da função que se pretende que seja realizada e deve ser um dos apresentados na Tabela 4.15.

Tabela 4.15 - Código de funções e respectivos dados

Código da Função	Dados
01	Coil Status
02	Input Status
03	Holding Registers

04	Input Registers

O campo *Address* define o ponto a endereçar e deve estar entre 1-65535. O campo *Length* define o comprimento, ou seja, o número de registos a ser lido – este valor deve estar entre 1-256.

Se o *PollModbus()* devolve zero, a mensagem foi enviada correctamente através da ligação e a aplicação receberá mais tarde um evento de *SlaveResponse*. De notar que o evento de *SlaveResponse* é disparado, mesmo que a mensagem demore mais tempo que o intervalo definido para o *time-out*. Caso o *PollModbus()* retorne um valor diferente de zero, o evento *SlaveResponse* não é disparado. O valor devolvido é então analisado pois trata-se de um código de erro, Tabela 4.16, que será processado de forma a identificar qual a falha ocorrida.

Tabela 4.16 – Tabela de códigos de erros e respectivas descrições

Código do erro	Descrição
0	Ok
<255	Slave Device Exception Response
256	Invalid Connection Handle
257	Message Overrun
258	Invalid Point Address
259	Invalid Slave Node Address
260	Invalid Length
261	Unsupported Modbus Command
263	Slave Device Time-out
264	Invalid Transmission Mode
265	Invalid CRC in Slave Response
266	Connection Not Established
267	Invalid Slave Response
272	Invalid Modbus/TCP command

No exemplo apresentado, pretende-se ler o valor contido nos 10 registos do escravo 1, começando no registo com endereço 100.

Após a aplicação iniciar um pedido de leitura a um dispositivo escravo, será disparado um evento de *SlaveResponse*, quando o resultado da mensagem estiver apto a ser lido.

ReadResults(), Tabela 4.17, é o método que devolve os resultados da última mensagem transaccionada na ligação.

Tabela 4.17 – Método ReadResults

Função	ReadResults
Parâmetros de entrada	Long ConnectionHandle, Short SlaveNodeAddress, Short ModbusFunctionCode, Long Address, Long *pData
Parâmetros de saída	Short Status
Exemplo	Status =mbMasterV71.ReadResults (iHandle,1,3,100+i,MyData(i))

Os parâmetros *SlaveNodeAddress* e *ModbusFunctionCode* deverão ser os mesmos que foram transmitidos na mensagem com o pedido de leitura (*PollModbus()*). O mesmo deverá acontecer relativamente ao endereço (*Address*) e ao número de registos (*Length*).

No exemplo apresentado, estão a ser lidos os resultados do *PollModbus* efectuado anteriormente. Os dados são lidos e, armazenados um por um no vector *MyData*. De notar que os campos referentes à ligação, endereço do escravo, código da função, número de registos e ser lidos e quantidade foram conservados desde o pedido de leitura *PollModbus()* – isto tem de acontecer sempre.

O método *ReadResults()* devolverá um código de erro, no parâmetro Status, igual a zero, caso a mensagem tenha sido transmitida correctamente e, os resultados apropriados, tenham sido recebidos. Se a recepção da mensagem demora mais do que o tempo estabelecido para o *timeout*, ou ocorreu um outro tipo de incorrecção, é gerado um código de erro que será analisado e identificado.

O ciclo de leitura de mensagens através deste método pode ser observado na Figura 4.23. De notar que este ciclo contém os valores referentes ao exemplo presente no quadro da função.

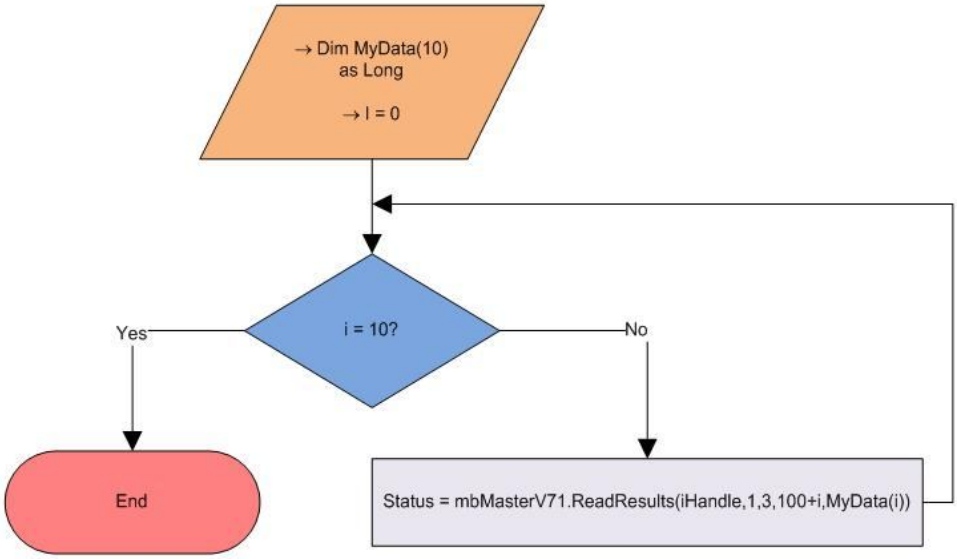


Figura 4.23 – Ciclo de leitura utilizando o método ReadResults()

Este método, *FillWriteBuffer*, Tabela 4.18, prepara um buffer de informação interno para ser transmitido ao escravo Modbus. A informação é escrita no buffer, valor por valor. Cada valor representa somente um registo ou uma *coil*.

Tabela 4.18 – Método FillWriteBuffer

Função	FillWriteBuffer
Parâmetros de entrada	Long ConnectionHandle, Short Index, Long Data
Parâmetros de saída	Short

Uma vez preenchido, o buffer está pronto a ser transmitido ao escravo designado através do método *WriteModbus()*, Tabela 4.19.

Tabela 4.19 – Método WriteModbus

Função	WriteModbus
Parâmetros de entrada	Long ConnectionHandle, Short SlaveNodeAddress, Short ModbusFunctionCode, Long Address, Short Length
Parâmetros de saída	Short Status
Exemplo	mbmasterV71.FillWriteBuffer(iHandle,0,1234) Status =mbMasterV71.WriteModbus (iHandle,1,6,100,1)

O método *WriteModbus* inicia um pedido de escrita ao escravo designado. O parâmetro *ConnectionHandle*, que define a ligação, deve ser igual ao devolvido pelo método *ConnectSerial()*, *DialTAPIDevice()* ou *ConnectModbusTCP()*.

O parâmetro *SlaveNodeAddress* define o escravo Modbus e, o *ModbusFunctionCode* deverá tomar um dos seguintes valores:

Código da Função	Dados
05	Single Coil Write
06	Single Register Write
15	Multiple Coil Write
16	Multiple Register Write

O campo *Address* especifica o ponto onde começará a escrita e o número de valores a escrever é especificado pelo campo *Length*. De ressaltar que a informação a escrever tem de ser previamente processada no método *FillWriteBuffer()*.

No exemplo apresentado, verifica-se a preparação do buffer com a informação a escrever (1234), escravo (1) e endereço (100). Só depois é dada a ordem de escrita nos registos do escravo designado.

Depois que a aplicação inicie uma mensagem de escrita para um escravo, é disparado um evento de *SlaveResponse* quando os resultados da mensagem estão disponíveis para leitura. O método *WriteResponse()*, Tabela 4.20, é pois o responsável pela leitura dos resultados provenientes do *WriteModbus()*.

Tabela 4.20 – Método WriteResponse

Função	WriteResponse
Parâmetros de entrada	Long ConnectionHandle, Short SlaveNodeAddress, Short ModbusFunctionCode, Long Address, Short Length
Parâmetros de saída	Short Status
Exemplo	Status =mbMasterV71.WriteResponse (iHandle,1,6,100,1)

Os parâmetros relativos ao endereço do escravo, código da função, endereço de escrita/leitura e quantidade de registos a ler têm de ser mantidos desde a execução do método *WriteModbus()*.

O *WriteResponse()* devolverá um código de erro igual a zero na variável *Status* caso a mensagem tenha sido correctamente transmitida ao escravo e os resultados apropriados tenham sido recebidos. No caso de ocorrer um *timeout* ou uma falha, será emitido um código de erro que será depois analisado e identificado.

O método *FillUserMsgBuffer*, Tabela 4.21, é designado para preparar um buffer interno de informação que, será posteriormente enviado ao dispositivo escravo.

Tabela 4.21 – Método FillUserMsgBuffer

Função	FillUserMsgBuffer
Parâmetros de entrada	Long ConnectionHandle, Short Index, Short Data
Exemplo	mbMasterV71.FillUserMsgBuffer (iHandle,0,1)

A informação tem de ser escrita no buffer, valor por valor, representado cada um deles um único byte. A aplicação é responsável por fornecer apenas os bytes de dados que compõem a mensagem a ser transmitida. Quando a mensagem está pronta a ser enviada, é-lhe acrescentado o correspondente checksum. Uma vez completo, o buffer está pronto a ser transmitido através do método *SendUserMsg()*, Tabela 4.22.

Tabela 4.22 – Método SendUserMsg

Função	SendUserMsg
Parâmetros de entrada	Long ConnectionHandle, Short Length
Parâmetros de saída	Short Status
Exemplo	mbMasterV71.FillUserMsgBuffer (iHandle,0,1) mbMasterV71.FillUserMsgBuffer (iHandle,1,3) mbMasterV71.FillUserMsgBuffer (iHandle,2,0) mbMasterV71.FillUserMsgBuffer (iHandle,3,1) mbMasterV71.FillUserMsgBuffer (iHandle,4,0) mbMasterV71.FillUserMsgBuffer (iHandle,5,10) Status =mbMasterV71.SendUserMsg (iHandle,6)

Este método inicia um pedido de escrita à ligação designada. *ConnectionHandle* define a ligação e deverá ser igual à devolvida pelos métodos *ConnectSerial(), DialTAPIDevice()* ou *ConnectModbusTCP*. O *checksum* (soma de verificação) é depois calculado e adicionado à mensagem. Se a ligação estiver no modo de transmissão ASCII, o caractér vírgula (,) será colocado na cabeça da mensagem e ser-lhe-á atribuída a respectiva combinação CRLF no final.

No exemplo apresentado, é possível observar o que foi descrito atrás. Assim, começa-se por construir o buffer usando o método *FillUserMsgBuffer*. Seguidamente, através do método *SendUserMsg*, envia-se a mensagem. Pode-se ainda verificar que a

mensagem em questão é composta por 6 bytes. Concluindo, após esta operação, será escrita a *string* "01030001000A", seguida do CRC.

Depois de uma aplicação iniciar uma mensagem de utilizador para um escravo Modbus, é disparado um evento de *UserMsgResponse* - quando os resultados estão prontos a ser lidos. Os parâmetros contidos nesse evento caracterizam a ligação e o número de bytes recebidos. Assim sendo, este método – *ReadUserMsgResponse*, Tabela 4.23 - permite ler a *string* resultante do buffer recebido após um *SendUserMsg()*.

Tabela 4.23 – Método ReadUserMsgResponse

Função	ReadUserMsgResponse
Parâmetros de entrada	Long ConnectionHandle, Short Index, short *pData
Parâmetros de saída	Short Status
Exemplo	Status =mbMasterV71.ReadUserMesgResponse(iHandle,0,Value)

Posto isto, tendo executado todas as ligações e configurações necessárias à comunicação, especificado o controlador, os tipos de mensagens que se podem enviar, como são recebidas e lidas as respostas, é altura de observar a aplicação implementada.

4.5 Exemplo de Aplicação

A rede implementada em conjunto com o software desenvolvido permite monitorizar o consumo energético em diversos pontos espalhados através de um ou vários complexos.

Esta aplicação, Figura 4.24, foi implementada e testada no Campus da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Pretende-se com isso, recolher dados relativamente aos consumos energéticos diários dos departamentos, traçar diagramas de carga, criar registos de consumos, facturação separada, identificar picos de energia e possíveis falhas.

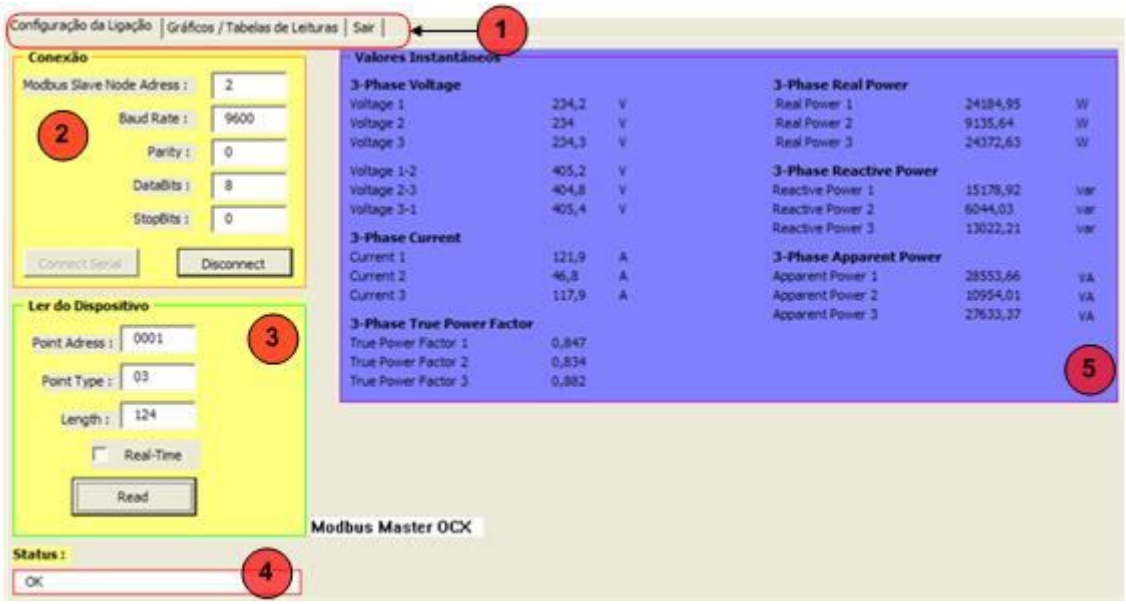


Figura 4.24 – Aplicação implementada

Na figura atrás apresentada estão marcados alguns pontos que se explicam seguidamente.

O ponto 1 é a barra de menus. Aqui o utilizador pode navegar entre o painel de configuração da ligação, criação de gráficos, elaboração de tabelas e sair da aplicação.

O ponto 2 é relativo aos parâmetros da ligação. Aqui é possível definir o baud rate da ligação, a paridade, número de bits de paragem, número de bits de informação e ainda identificar o dispositivo com quem se pretende comunicar. Neste painel existem ainda dois botões que servem respectivamente para estabelecer a ligação e para quebrá-la.

Depois de estabelecidos os parâmetros da ligação (ponto 2), é necessário definir a função que se pretende que o escravo Modbus execute, qual o endereço de partida onde se vai escrever ou ler informação e quantos registos serão lidos/escritos. Aqui pode-se optar por uma leitura de cada vez, bastando para isso carregar no botão *Read*. Caso se pretenda observar os valores em tempo real, basta marcar a *checkbox* "*Real-Time*" e carregar no botão de *Read*. Tudo isto é disponibilizado no painel 3.

A barra de "Status", marcada com o número 4, informa o utilizador relativamente ao estado das operações que este pretende ver executadas. Por exemplo, informa se a ligação foi estabelecida ou não, se a função que se pretende executar teve ou não sucesso, se existem falhas, etc.

Finalmente, no painel 5, é possível observar os valores lidos dos dispositivos Modbus. São apresentados os valores relativos à tensão e corrente, em cada fase, do sistema trifásico que está a ser monitorado. É possível ainda observar-se os valores das potências activa, reactiva e aparente, bem como o factor de potência da instalação.

4.5.1 Tratamento de Dados

Apresentada interface da aplicação desenvolvida, é hora de verificar qual o código por detrás desta.

Assim sendo, nesta secção dar-se-á ênfase ao programa desenvolvido, onde são armazenados os dados das mensagens, como chegam até à aplicação e quais as formatações que têm de sofrer para que sejam correctamente observados. No final serão apresentados alguns resultados que servirão para comprovar o que tem vindo a ser mencionado ao longo deste exercício.

Como em qualquer tipo de comunicação, em primeiro lugar é necessário definir uma ligação e quais os parâmetros que a caracterizam. A aplicação aqui apresentada não foge à regra. O excerto de código, Figura 4.25, apresentado seguidamente mostra como é estabelecida a ligação.

```
Private Sub ConnectSerial_Click()
    MbMasterV71.BaudRate = 9600             '9600 Baud
    MbMasterV71.PARITY = 0                  '0=NOPARITY, 1=ODDPARITY, 2=EVENPARITY, 3=MARKPARITY, 4=SPACEPARITY
    MbMasterV71.DataBits = 8                '8 DataBits
    MbMasterV71.StopBits = 0                '0=ONESTOPBIT, 1=ONE5STOPBITS, 2=TWOSTOPBITS
    MbMasterV71.TimeOut = 5000              '5000 msec
    MbMasterV71.TransmissionMode = 1        '0=ASCII, 1=RTU
    MyHandle = MbMasterV71.ConnectSerial(1) 'Ligar à porta COM 1
    If MyHandle > 0 Then                    'MyHandle funciona como uma flag que informa o estado da conexão
        ConnectSerial.Enabled = False       'Desactiva o botão de conexão na form
        Desconectar.Enabled = True          'Activa o botão para desconectar a ligação
        Status.Text = "Connected"           'Informa que a conexão foi bem sucedida
        READMODBUS.Enabled = True           'Activa botão de leitura
    Else                                    'A Conexão Falhou!
        Status.Text = "Not Connected"       'Informa que a conexão foi mal sucedida
    End If
End Sub
```

Figura 4.25 – Função ConnectSerial

Como é possível observar, a ligação é definida com um *baud rate* de 9600, paridade nula, 8 *databits* e nenhum *stop bit*. Para o *timeout* foi escolhido o valor de 5000 mseg e para o modo de transmissão foi escolhido o Modbus RTU.

Estes valores poderão ser alterados mas, deverão ser os mesmos na aplicação e no dispositivo escravo para que não existam falhas nem erros na conexão.

É ainda possível verificar que a ligação é estabelecida através da porta COM 1 e, se a "manivela" de ligação (*MyHandle*) for devolvida com um valor superior a zero, então a conexão é estabelecida.

Caso a ligação não tenha sido estabelecida, *MyHandle* retornará -1, Figura 4.26, tal como especificado aquando da apresentação dos métodos em 4.3.3.

```
Private Sub ConnectSerial_Click()
    MbMasterV71.BaudRate = 9600         '9600 Baud
    MbMasterV71.PARITY = 0               '0=NOPARITY, 1=ODDPARITY, 2=EVENPARITY, 3=MARKPARITY, 4=SPACEPARITY
    MbMasterV71.DataBits = 8             '8 DataBits
    MbMasterV71.StopBits = 0             '0=ONESTOPBIT, 1=ONE5STOPBITS, 2=TWOSTOPBITS
    MbMasterV71.TimeOut = 5000           '2000 msec
    MbMasterV71.TransmissionMode = 1     '0=ASCII, 1=RTU
    MyHandle = MbMasterV71.ConnectSerial(1)  'Ligar à porta COM 1
    If MyHandle > 0 Then                 'MyHandle funciona como uma flag que informa o estado da conexão
        MbMasterV71.Serial.Enabled = False   'Desactiva o botão de conexão na form
        Desconectar.Enabled = True           'Activa o botão para desconectar a ligação
        Status.Text = "Connected"            'Informa que a conexão foi bem sucedida
        READMODBUS.Enabled = True            'Activa botão de leitura
    Else                                     'A Conexão Falhou!
        Status.Text = "Not Connected"        'Informa que a conexão foi mal sucedida
    End If
End Sub
```

Figura 4.26 – Falha na ligação

Pode ainda dar-se o facto de o utilizador tentar emitir um pedido de leitura, através do botão "*Read*" da aplicação sem ter previamente efectuado a conexão. Se isso acontecer, haverá um erro, Figura 4.27. A aplicação informará então o utilizador desse mesmo erro, colocando a descrição na barra de *status*.

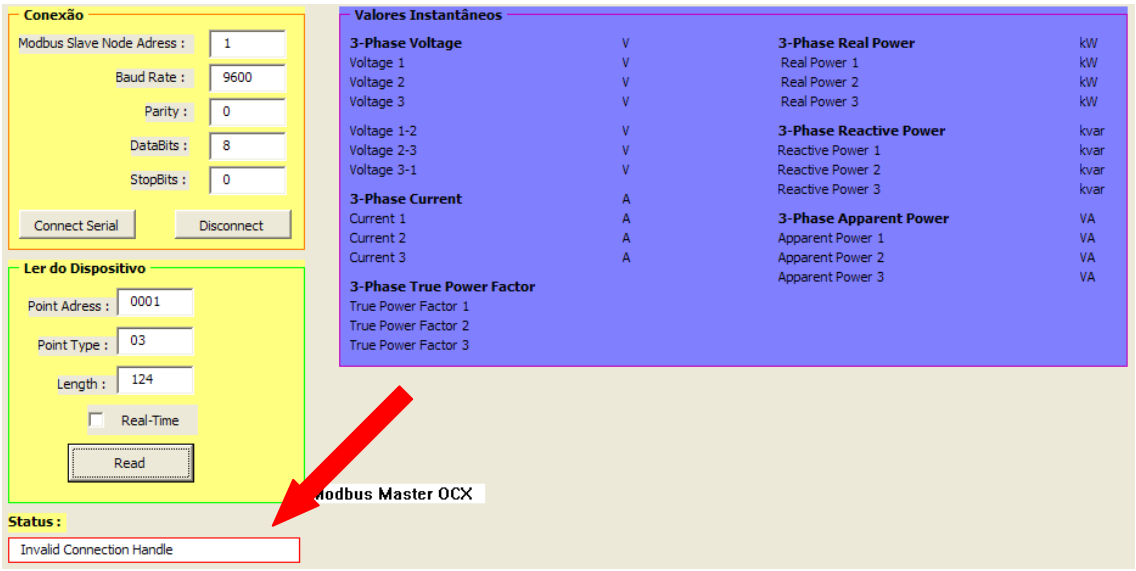


Figura 4.27 – Erro de ligação

Mas, com é esse erro processado e identificado? Pois bem, tal como mencionado atempadamente, o controlador começa por gerar um código de erro, Figura 4.28.

```
    If MyStatus = 0 Then            ' Verifica se o pedido foi enviado
        MyStatus = 256ext = "Busy"
    Else
        show_status (MyStatus)
    End If
```

Figura 4.28 – Código de erro emitido pelo controlador mbMasterV71

Seguidamente, é percorrida a lista de erros possíveis e, esse mesmo erro é identificado, Figura 4.29.

```
Public Sub show_status(ErrCode As Integer)

    If (ErrCode = 0) Then
        Status.Text = "OK"
    ElseIf (ErrCode < 255) Then
        Status.Text = "Slave Device Exception Response"
    ElseIf (ErrCode = 256) Then
        Status.Text = "Invalid Connection Handle"
    ElseIf (ErrCode = 257) Then
        Status.Text = "Message Overrun"
    ElseIf (ErrCode = 258) Then
```

Figura 4.29 – Identificação do erro com código 256

Pode ainda dar-se o facto de o utilizador estabelecer modos de transmissão diferentes na aplicação e no dispositivo escravo. Este engano levará ao aparecimento de um novo erro pois o escravo não conseguirá compreender a mensagem recebida e, não enviará resposta ao mestre, dando origem ao erro "*Slave Device Time-Out*", Figura 4.30. Ou seja, o escravo fica em silêncio e, quando passa o tempo de espera definido, surge a mensagem de erro.

Figura 4.30 – Erro Slave Device Time-Out

Quando o utilizador pretende obter informação de um escravo, terá de indicar qual o endereço a partir do qual quer ler informação, o número de registos a ser lido, e a função que deseja ver executada. Posto isto e, identificado o dispositivo ao qual se quer referir, basta carregar no botão "*Read*", Figura 4.31.

Ao realizar estes passos, o utilizador estará simplesmente a executar um *PollModbus* – método já apresentado.

```
Private Sub READMODBUS_Click()
Dim MyData As Long

    Slave = NODEADDRESS.Text
    Cmd = POINTTYPE.Text
    Address = READADDRESS.Text
    Length = READLENGTH.Text
    MyStatus = MbMasterV71.PollModbus(MyHandle, Slave, Cmd, Address, Length)

    If MyStatus = 0 Then
        Status.Text = "Busy"
    Else
        show_status (MyStatus)
    End If

End Sub
```

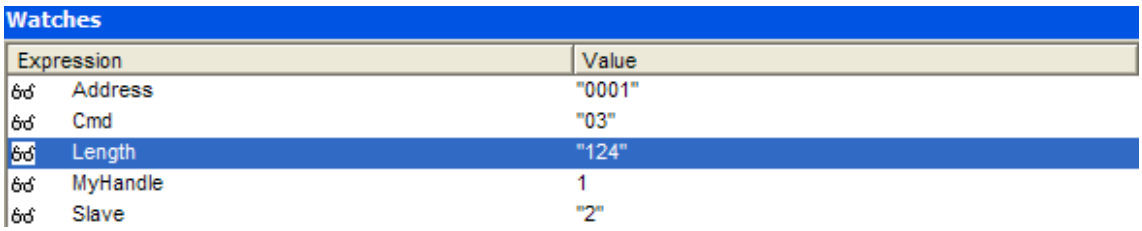
Figura 4.31 – Função associada ao botão "Read"

Nesta função é possível verificar o carregamento da informação necessária ao envio da mensagem de *PollModbus*. Essa informação é proveniente das caixas de texto presentes na interface de utilizador, tal como assinalado na Figura 4.32.

Figura 4.32 – Interface de utilizador evidenciando os parâmetros do dispositivo

Seguidamente é executado o método *PollModbus*. Este método ficará com o seguinte aspecto:

```
    MyStatus = MbMasterV71.PollModbus(MyHandle, Slave, Cmd, Address, Length)
```



Resumindo, este *PollModbus* é um pedido de leitura (*Cmd* = 03) ao escravo identificado pelo endereço 2 (*Slave* = 2), começando a leitura no endereço 1 (*Address* = 1) e lendo um total de 124 registos (*Length* = 124).

Como resultado desta acção, e no caso de não ocorrerem falhas, será disparado um evento de *SlaveResponse*, indicando ao mestre que a resposta do escravo está disponível para leitura. (De notar que no caso de existir uma falha, o evento de *SlaveResponse* será lançado na mesma mas, contendo um código de erro identificativo da falha ocorrida.)

Assim, utilizando o método *ReadResults*, poder-se-á ler a informação enviada pelo escravo, Figura 4.33. Atente-se que os parâmetros relativos ao endereço, função pedida ao escravo, endereço de leitura e quantidade de registos mantêm-se inalteráveis desde a execução do *PollModbus* – isto tem de acontecer sempre.

```
Private Sub MbMasterV71_SlaveReadResponse(ByVal hConnect As Long)

    Slave = NODEADDRESS.Text
    Cmd = POINTTYPE.Text
    Address = READADDRESS.Text
    Length = READLENGTH.Text

    For i = 0 To Length - 1
        MyStatus = MbMasterV71.ReadResults(hConnect, Slave, Cmd, Address + i, MyData)
        MyArray(i) = MyData
        show_status (MyStatus)
        If MyStatus = 0 Then
        End If
    Next i

    For i = 0 To 124
        Folha2.Cells(1, 1 + i) = MyArray(i)
    Next i
End Sub
```

Figura 4.33 – Função SlaveReadResponse

Pelo excerto de código acima representado, Figura 4.33, verifica-se que é necessário utilizar um ciclo para efectuar a leitura da informação retornada pelo escravo (ciclo representado a vermelho). As razões para tal acontecer foram já apresentadas em 4.3.3.

Seguidamente ao ciclo de leitura, é implementado um ciclo de escrita (Ciclo representado a verde) para uma folha Excel, onde ficarão registados os valores.

Como resultado destas operações, o vector *MyArray* tomará o aspecto apresentado na Figura 4.34.

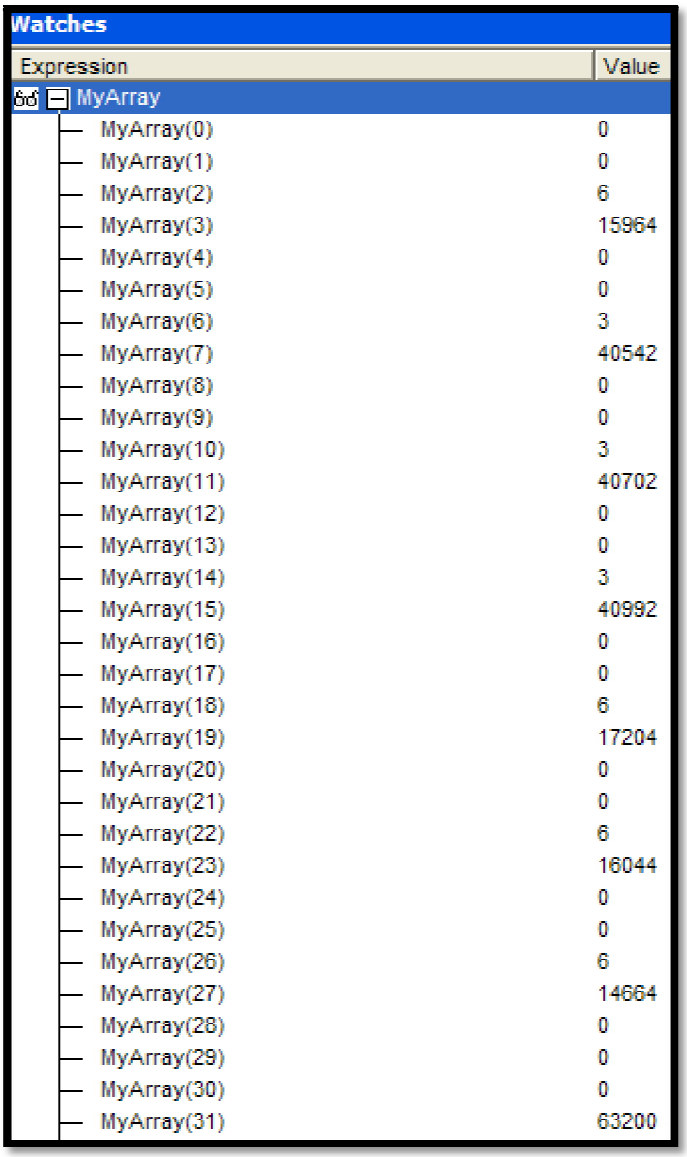


Figura 4.34 – Vector MyArray depois de uma acção de leitura

No que respeita à folha de Excel obtém-se algo como o que é apresentado na Figura 4.35.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	0	0	6	15964	0	0	3	40542	0	0	3	40702	0	0	3	40992	0	0	6	17204

Figura 4.35 – Dados do MyArray escritos numa folha Excel

Olhando com atenção, os valores apresentados pelo *MyArray* não estão correctamente formatados. Assim sendo, existe a necessidade de efectuar algumas conversões de modo a termos os valores correctos.

Segundo as especificações do fabricante dos medidores UPT210, cada valor de tensão corrente, potência, etc., é representado no formato "*unsigned*" por dois números, como por exemplo 6 e 15964, Figura 4.35. Assim, existem algumas alterações que devem ser realizadas de forma a obter os resultados por que correctos - tais modificações serão feitas dentro da função *SlaveReadResponse* que, tomará o seguinte aspecto, Figura 4.36:

```
Private Sub MbMasterV71_SlaveReadResponse(ByVal hConnect As Long)

    Slave = NODEADDRESS.Text
    Cmd = POINTTYPE.Text
    Address = READADDRESS.Text
    Length = READLENGTH.Text

    ' CICLO DE LEITURA
    For i = 0 To Length - 1
        MyStatus = MbMasterV71.ReadResults(hConnect, Slave, Cmd, Address + i, MyData)
        show_status (MyStatus)
        MyArray(i) = (DecToBin(CStr(MyData)))
        If MyStatus = 0 Then
        End If
    Next i

    'CARREGA OS VALORES CONVERTIDOS DE BINARIO PARA DECIMAL PARA O NOVO ARRAY
    j = 0
    For i = 0 To 124 Step 2
    NewArray(j) = 0.001 * (BinToDec(MyArray(i) & MyArray(i + 1)))
    j = j + 1
    Next i

    'ESCREVE NA FOLHA 2 O ARRAY COM OS VALORES CORRECTOS
    For i = 0 To 124 '124
        Folha2.Cells(1, 1 + i) = (NewArray(i))
    Next i

End Sub
```

Figura 4.36 – Aspecto final da função SlaveReadResponse

No ciclo de leitura, Figura 4.36, é acrescentada uma conversão DecToBin que vai converter os valores presentes na variável *MyData* de *unsigned* para binário. Tendo os valores em binário, acoplam-se utilizando a expressão *MyArray(i)&MyArray(i+1)*. Posto isto, resta converter novamente os valores de binário para decimal - Esta conversão é feita através da função BinToDec.

Finalmente, um pequeno ajuste em relação às unidades. Multiplicam-se todos os valores por 0.001 para ficarem na unidade fundamental e, obtêm-se os valores pretendidos (Figura 4.37).

Data	Hora	Tensão da fase 1 (V)	Tensão da fase 2 (V)	Tensão da fase 3 (V)	Tensao Simples das 3 fases (V)	Tensão 1-2 (V)	Tensão 2-3 (V)	Tensão 3-1 (V)	Tensao Composta das 3 fases (V)	Corrente da fase 1 (A)	Corrente da fase 2 (A)	Corrente da fase 3 (A)	Corrente das 3 fases (A)
25-06-2010	0:01:21	231,0	231,6	231,9	231,5	399,6	400,7	401,2	400,5	43,4	50,7	33,4	42,5
25-06-2010	0:01:53	230,6	231,3	231,5	231,1	399,0	400,1	400,5	399,9	43,4	50,7	33,6	42,6
25-06-2010	0:02:24	230,5	231,3	231,2	231,0	398,8	400,2	400,0	399,7	43,4	50,6	33,4	42,5
25-06-2010	0:02:56	230,3	231,2	231,3	231,0	398,5	400,0	400,2	399,5	43,7	50,9	33,6	42,7
25-06-2010	0:03:28	232,4	233,1	233,4	233,0	402,1	403,3	403,7	403,0	43,7	50,7	33,4	42,6
25-06-2010	0:04:00	232,4	233,0	233,3	232,9	402,0	403,1	403,7	402,9	43,4	50,1	33,4	42,3
25-06-2010	0:04:32	232,5	233,2	233,5	233,0	402,1	403,3	403,9	403,1	43,5	50,4	33,4	42,5
25-06-2010	0:05:04	232,4	233,0	233,2	232,9	402,0	403,1	403,4	402,8	47,0	50,4	44,2	47,2
25-06-2010	0:05:36	232,6	233,0	233,3	233,0	402,4	403,1	403,6	403,0	45,8	50,6	42,4	46,2
25-06-2010	0:06:07	233,0	233,2	233,4	233,2	403,1	403,5	403,7	403,4	45,8	50,6	42,1	46,1
25-06-2010	0:06:39	232,8	233,3	233,8	233,3	402,7	403,6	404,5	403,6	43,5	52,5	33,3	43,1
25-06-2010	0:07:11	233,3	233,8	234,1	233,7	403,5	404,5	405,1	404,4	42,9	50,6	33,3	42,2
25-06-2010	0:07:43	233,6	234,1	234,4	234,0	404,1	404,9	405,4	404,8	42,9	50,4	33,4	42,2
25-06-2010	0:08:15	233,4	234,1	234,5	234,0	403,8	404,9	405,6	404,8	42,9	50,2	33,6	42,2
25-06-2010	0:08:47	233,4	234,1	234,6	234,0	403,8	404,9	405,8	404,8	42,9	50,2	33,6	42,2
25-06-2010	0:09:23	233,4	234,0	234,4	233,9	403,8	404,8	405,5	404,7	42,9	50,2	33,6	42,2
25-06-2010	0:09:55	233,2	233,8	234,2	233,7	403,4	404,5	405,2	404,4	43,2	51,0	33,6	42,6
25-06-2010	0:10:27	232,8	233,5	233,9	233,4	402,7	403,9	404,7	403,7	43,0	50,4	33,6	42,3
25-06-2010	0:11:00	232,8	233,7	234,1	233,5	402,7	404,4	404,9	404,0	42,7	49,8	33,6	42,0
25-06-2010	0:11:32	233,5	234,0	234,5	234,0	404,0	404,9	405,6	404,8	42,7	49,9	33,6	42,1
25-06-2010	0:12:05	233,8	234,1	234,3	234,1	404,5	404,9	405,3	404,9	42,9	49,9	33,4	42,1
25-06-2010	0:12:37	234,0	234,4	234,5	234,3	404,9	405,5	405,7	405,4	48,5	52,0	42,7	47,7
25-06-2010	0:13:09	233,9	234,4	234,4	234,3	404,7	405,5	405,5	405,3	45,9	50,1	42,6	46,2
25-06-2010	0:13:42	233,7	234,5	234,6	234,2	404,2	405,6	405,8	405,2	43,5	50,1	33,8	42,5
25-06-2010	0:14:14	233,8	234,5	234,8	234,3	404,5	405,7	406,1	405,4	44,0	51,4	33,8	43,0
25-06-2010	0:14:47	233,4	234,2	234,6	234,1	403,8	405,2	405,8	404,9	43,7	50,1	33,8	42,5
25-06-2010	0:15:19	233,4	234,2	234,6	234,1	403,7	405,2	405,8	404,9	44,0	51,4	33,8	43,0

Figura 4.37 – Valores correctamente formatados e na unidade fundamental (Retirado da folha de armazenamento do Excel)

Foram ainda implementadas funções que dão a possibilidade de fazer gráficos, guardar valores, imprimir tabelas ou enviar os ficheiros através de email.

Apesar do Excel possuir um número limitado de linhas e colunas, esse limite nunca é ultrapassado pela solução aqui apresentada pois no final de cada dia, os dados são guardados e as folhas de registo intermédio são apagadas.

Nesta secção ficou demonstrado como se adquirem os dados e quais as transformações a que estão sujeitos antes de serem apresentados ao utilizador.

Seguidamente, dar-se-á ênfase a alguns resultados recolhidos após a implementação deste sistema na Faculdade de Ciências e Tecnologia.

4.5.2 Dados de Consumo Energético

Ao monitorizar o consumo de energia, as empresas obtém respostas a uma série de importantes questões, tais como:

- Quais as máquinas ou departamentos que utilizam mais energia?
- O que está a causar picos de consumo?
- O que se pode dizer acerca do factor de potência?
- Que consumos anormais ocorreram e quando?

Tendo estas questões em mente e, tendo acesso aos dados de consumo energético, torna-se possível gerar alarmes automáticos para os responsáveis pela gestão energética

quer seja via e-mail ou mesmo mensagens de texto que, permitirão rápidas intervenções e poupanças imediatas.

A utilização da monitorização de energia permite criar uma "cultura de consciência energética". Com a ferramenta desenvolvida obtém-se leituras rápidas e precisas, Figura 4.38, dos valores de tensão, corrente e factor de potência, tanto para um sistema monofásico como para um sistema trifásico.

Data	Hora	Tensao Simples (V)	Tensao Composta (V)	Corrente (A)	Potência Activa (W)	Factor de Potência	Potência Reactiva (VAR)	Potência Aparente (VA)
25-06-2010	0:01:21	231,5	400,5	42,5	28224,7	0,96	8363,2	29518,6
25-06-2010	0:01:53	231,1	399,9	42,6	28225,8	0,96	8335,5	29510,4
25-06-2010	0:02:24	231,0	399,7	42,5	28154,1	0,96	8244,3	29423,1
25-06-2010	0:02:56	231,0	399,5	42,7	28321,8	0,96	8308,5	29596,6
25-06-2010	0:03:28	233,0	403,0	42,6	28458,0	0,96	8481,8	29779,3
25-06-2010	0:04:00	232,9	402,9	42,3	28267,9	0,96	8328,8	29547,3
25-06-2010	0:04:32	233,0	403,1	42,5	28388,3	0,96	8369,0	29674,6
25-06-2010	0:05:04	232,9	402,8	47,2	31629,8	0,96	8985,2	32971,7
25-06-2010	0:05:36	233,0	403,0	46,2	31031,7	0,96	8667,8	32315,4
25-06-2010	0:06:07	233,2	403,4	46,1	30976,3	0,96	8676,0	32273,1
25-06-2010	0:06:39	233,3	403,6	43,1	28919,8	0,96	8263,7	30155,4
25-06-2010	0:07:11	233,7	404,4	42,2	28339,9	0,96	8268,8	29615,9
25-06-2010	0:07:43	234,0	404,8	42,2	28320,0	0,96	8528,1	29649,6
25-06-2010	0:08:15	234,0	404,8	42,2	28397,4	0,96	8222,4	29645,9
25-06-2010	0:08:47	234,0	404,8	42,2	28366,9	0,96	8391,5	29649,6
25-06-2010	0:09:23	233,9	404,7	42,2	28401,8	0,96	8099,3	29639,5
25-06-2010	0:09:55	233,7	404,4	42,6	28620,3	0,96	8302,6	29878,3
25-06-2010	0:10:27	233,4	403,7	42,3	28367,8	0,96	8277,2	29644,1
25-06-2010	0:11:00	233,5	404,0	42,0	28191,3	0,96	8139,7	29440,4
25-06-2010	0:11:32	234,0	404,8	42,1	28294,9	0,96	8160,0	29537,0
25-06-2010	0:12:05	234,1	404,9	42,1	28287,9	0,96	8271,0	29544,6
25-06-2010	0:12:37	234,3	405,4	47,7	32482,7	0,97	7240,0	33552,4

Figura 4.38 – Dados caracterizadores de uma rede eléctrica (Obtidos através da aplicação desenvolvida)

Com os dados recolhidos pode-se construir gráficos, tornando-se a análise mais intuitiva, Figura 4.39.

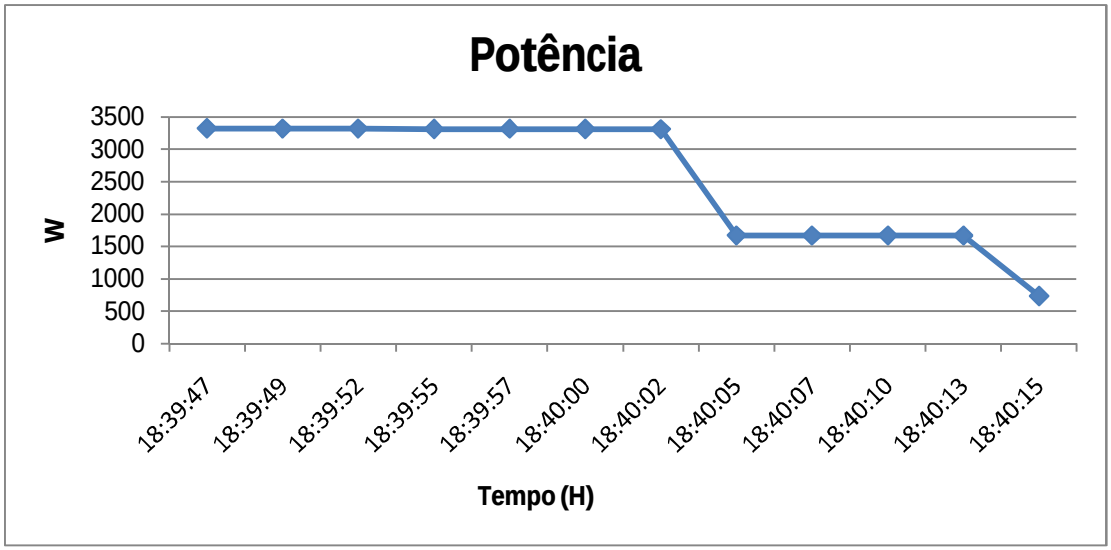


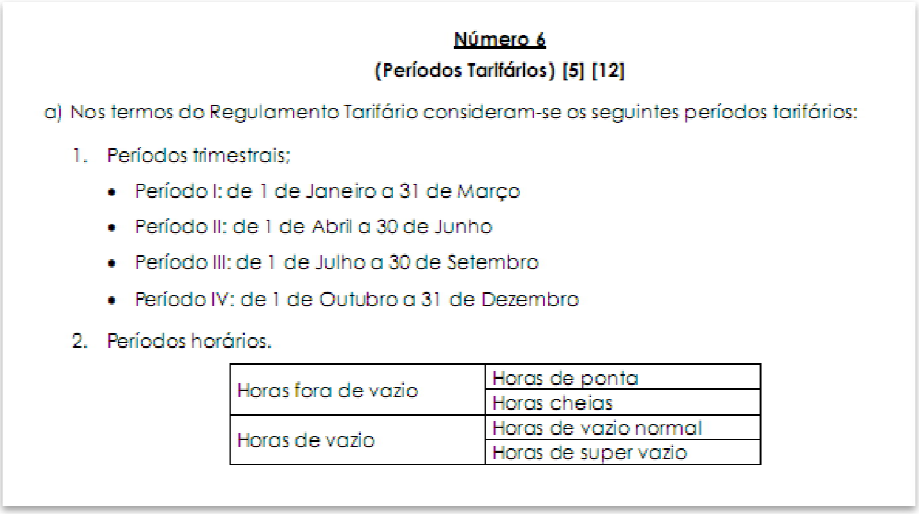
Figura 4.39 – Exemplo de um gráfico de potência obtido através de dados recolhidos pela aplicação

Tanto os dados recolhidos como os gráficos que deles advêm podem posteriormente ser combinados de forma a gerar relatórios. Os relatórios poderão ser de contagem –

mapeamento dos consumos por contador -, relatórios de consumidor, onde se poderão agrupar vários contadores existentes num dado departamento e, relatórios de histórico, onde seriam apresentados os dados para uma máquina específica, para um tipo de máquina ou para toda a fábrica.

Tendo em conta a hora do dia e a estação do ano, a empresa fornecedora de energia eléctrica, taxa a energia de forma diversa, Figura 4.40. Assim sendo, existem quatro períodos diários a ter em conta quando se fala no consumo energético diário:

- Super Vazio
- Cheias
- Ponta
- Vazio Normal



Número 6

(Períodos Tarifários) [5] [12]

a) Nos termos do Regulamento Tarifário consideram-se os seguintes períodos tarifários:

1. Períodos trimestrais:
   - Período I: de 1 de Janeiro a 31 de Março
   - Período II: de 1 de Abril a 30 de Junho
   - Período III: de 1 de Julho a 30 de Setembro
   - Período IV: de 1 de Outubro a 31 de Dezembro
2. Períodos horários.

Horas fora de vazio	Horas de ponta
	Horas cheias
Horas de vazio	Horas de vazio normal
	Horas de super vazio

Figura 4.40 – Períodos Tarifários da EDP

A título de curiosidade, os relatórios de consumo de energia estão bem documentados, pela indústria, em muitos países. Por exemplo, no Reino Unido, as empresas recebem subsídios governamentais se o sistema de monitorização de energia instalado também incluir um conjunto de relatórios bem delineados.

4.6 Resultados

O sistema até agora mencionado, foi utilizado para monitorar um painel fotovoltaico, um gerador eólico e, o quadro geral de baixa tensão do Departamento de Engenharia Electrotécnica. Seguidamente, apresentam-se alguns resultados, dando-se particular ênfase nos gráficos pois são mais elucidativos que uma grande quantidade de dados.

Na Figura 4.41 é apresentada a evolução da potência activa do Quadro Geral de Baixa Tensão presente no Departamento de Engenharia Electrotécnica. Os dados dizem respeito a um período de 24 horas correspondente ao dia 25 de Junho de 2010.

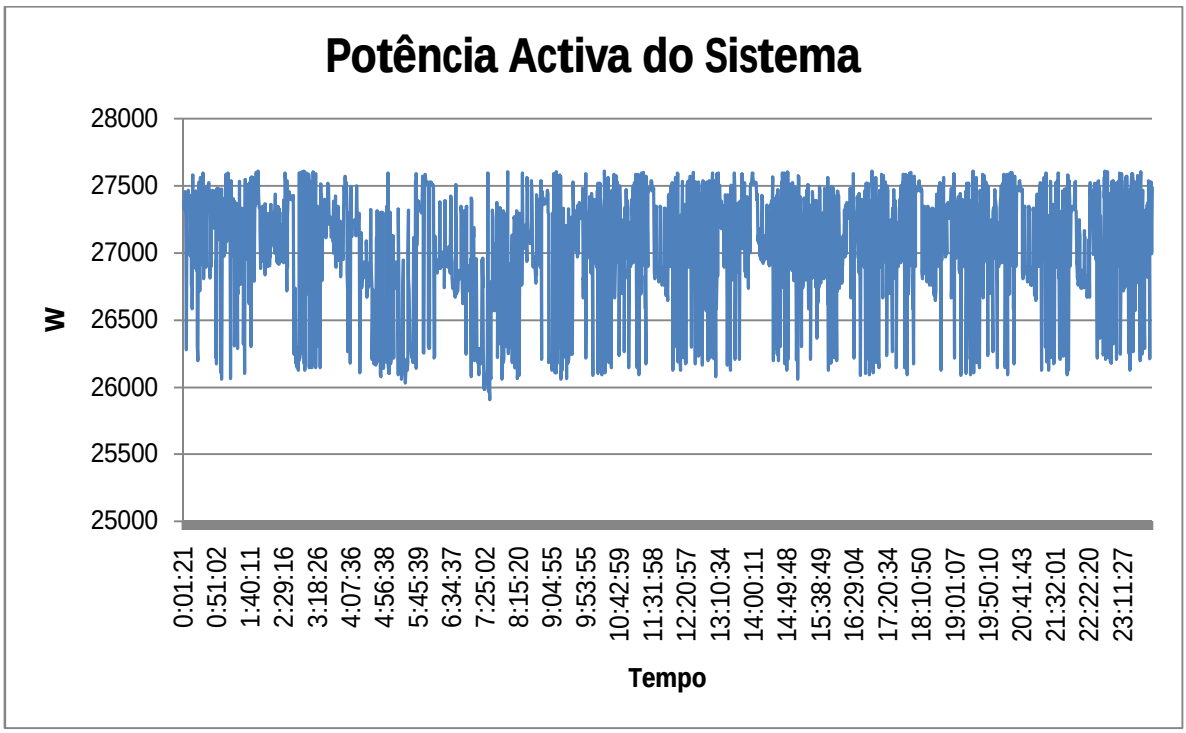


Figura 4.41 – Potência activa do QGBT no dia 25 de Junho de 2010

A potência reactiva do QGBT é apresentada na Figura 4.42. Os valores observados foram coram colhidos no mesmo dia (25 de Junho de 2010) e, novamente resolveu-se apresentar um período 24 horas, coincidente com o utilizado para a potência activa.

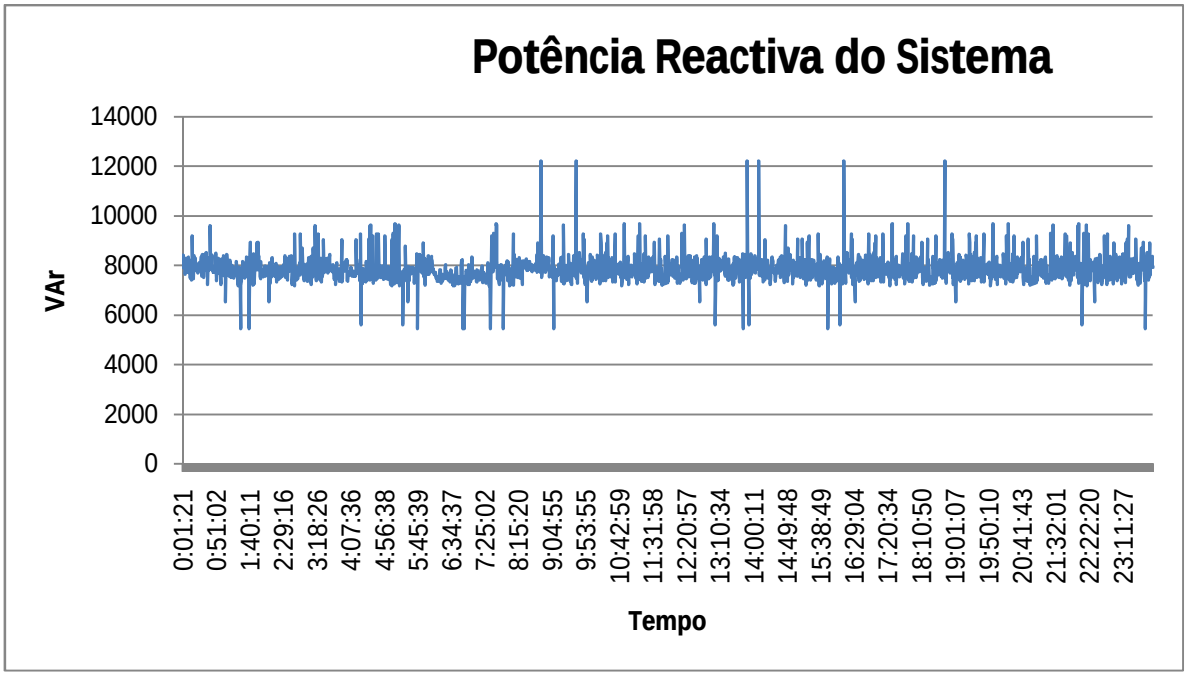


Figura 4.42 – Potência reactiva do QGBT no dia 25 de Junho de 2010

No mesmo período de tempo mencionado foi também observada a evolução da potência aparente, tal como mostrado na Figura 4.43.

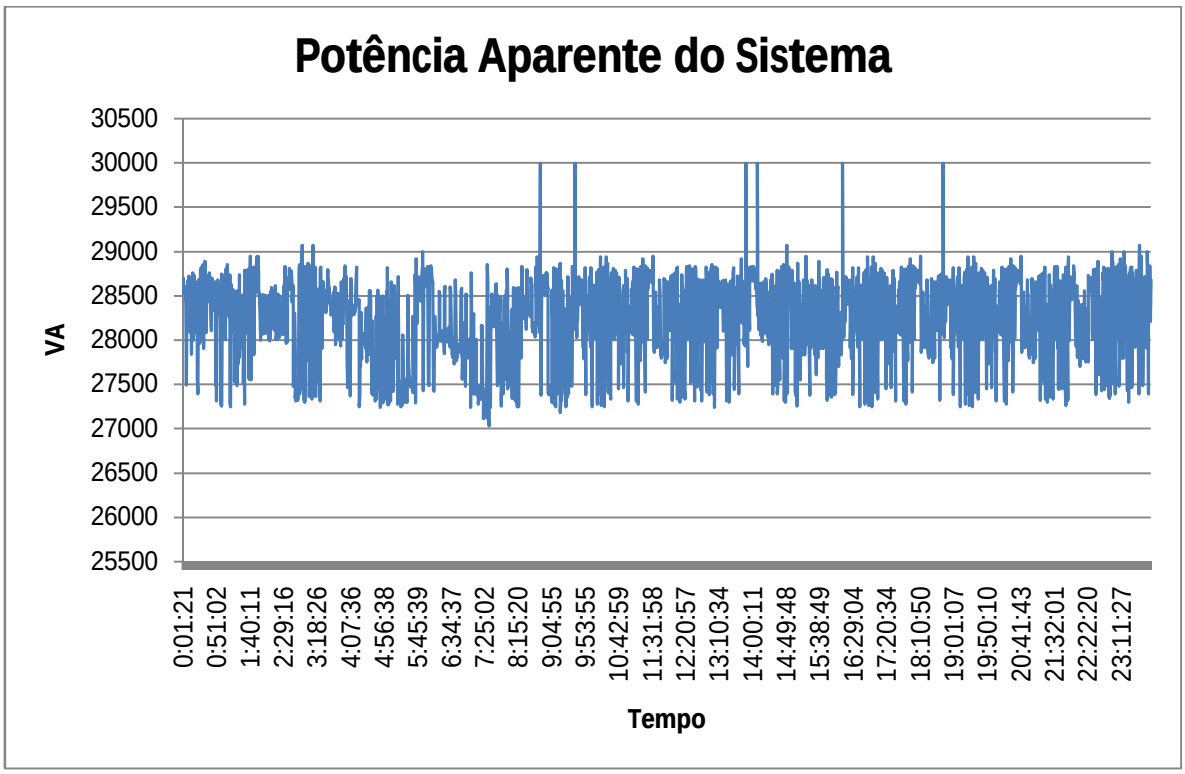


Figura 4.43 – Potência aparente do QGBT no dia 25 de Junho de 2010

O comportamento das tensões (simples e compostas) no QGBT é evidenciado na Figura 4.44.

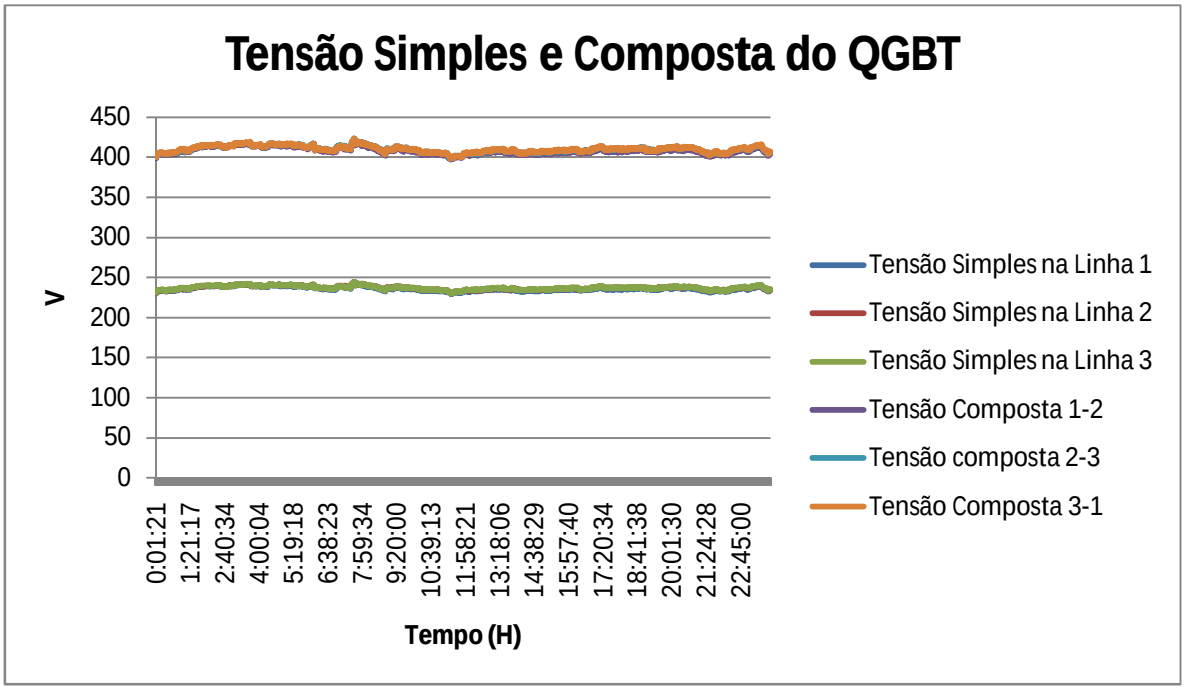


Figura 4.44 – Tensão Simples e Composta no QGBT no dia 25 de Junho de 2010

No que toca ao gerador eólico e painel fotovoltáico, deve-se ter em conta que se as correntes geradas forem muito baixas, essas não são contabilizadas devido à capacidade dos medidores UPT210. Assim sendo, se a corrente não é contabilizada, as potências aparecerão como nulas. Isto porque as potências são calculadas de forma indirecta através dos valores

das tensões das correntes e do factor de potência, tal como se poderá verificar pelas equações seguintes.

- Potência Activa

$$P = U_c I_c . \cos\phi ,$$

onde I_c é o valor eficaz da intensidade de corrente alternada sinusoidal, U_c é o valor da tensão sinusoidal e ϕ é o ângulo de fase ou desfasagem entre a tensão e a corrente. O termo $\cos\phi$ é denominado Factor de Potência. Se I_c está em amperes, U_c em volts, P estará em watts (W).

- Potência Reactiva

$$Q = U_c I_c . sen\phi ,$$

onde I_c é o valor eficaz da intensidade de corrente alternada sinusoidal, U_c é o valor da tensão sinusoidal e ϕ é o ângulo de fase ou desfasagem entre a tensão e a corrente. Se I_c está em amperes, U_c em volts, Q estará em VAr - Volt Ampère reativo. Na indústria eléctrica recomenda-se que todas as instalações tenham um factor de potência máximo, com o qual $sen\phi$ será mínimo e portanto a potência reactiva ou não útil será também mínima.

- Potência Aparente

pelo triângulo de potências, Figura 4.45, é fácil obter a expressão da potência aparente. Basta calcular a hipotenusa do triângulo de potências, donde:

$$S = \sqrt{P^2 + Q^2}$$

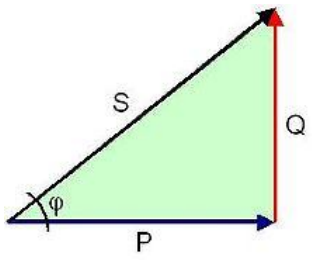


Figura 4.45 – Relação entre as potências Activa, Reactiva e Aparente

É com base no valor desta potência (medida em VA, Volt Ampère) que é feito o dimensionamento das cablagens e sistemas de protecção das instalações eléctricas. Na contratação do fornecimento de energia eléctrica é normalmente especificada a taxa de potência que depende da potência aparente máxima a ser disponibilizada pelo fornecedor. Mas, essa não é a potência trifásica e sim a monofásica. Para calcular a potência trifásica basta na mesma fórmula multiplicar também o resultado por raiz de três.

Capítulo 5 - Conclusões

Neste capítulo seguinte serão tecidas as considerações finais relativamente ao exercício que vem sendo exposto. Algumas sugestões relativamente a trabalhos futuros com base no sistema desenvolvido serão também apresentadas.

5.1 Síntese do trabalho e Considerações

Para uma correcta gestão dos meios produtivos e dos custos operacionais de uma qualquer empresa, tornou-se essencial o controlo do consumo de energia.

Um dos problemas que se coloca, em matéria de gestão de energia, em muitas empresas é a falta de informação disponível sobre os consumos energéticos dos processos.

É frequente encontrarem-se em muitas empresas, determinados equipamentos ou sectores responsáveis por uma grande parte do consumo global, sem que tenham contadores instalados, o que impossibilita a determinação dos respectivos consumos específicos bem como a detecção de situações de consumos anómalos.

Deste modo, e com o objectivo de se dispor de uma informação constantemente actualizada e de possibilitar o controlo dos consumos energéticos dos diferentes sectores ou equipamentos de uma empresa, foi desenvolvido um sistema de monitorização de consumos de energia denominado Energy Monitor o qual consiste, de uma forma geral, na monitorização constante dos consumos de energia, permitindo definir valores de referência em relação aos quais é possível detectar eventuais desvios e estabelecer, posteriormente, metas de redução dos consumos energéticos.

Ao monitorizar o consumo de energia, a empresa verifica: os departamentos ou máquinas mais dispendiosos, as flutuações do consumo de energia por máquina ou área produtiva e, consegue identificar a utilização errática (fora do padrão adequado) da energia.

A utilização da monitorização de energia cria uma "cultura de consciência energética" na empresa entre todos os colaboradores, constituindo-se como a ferramenta ideal para a empresa atingir os seus objectivos.

O Energy Monitor é uma ferramenta útil para a obtenção de dados e análise do consumo energético. Permite a visualização "*online"* de vários parâmetros energéticos (tensões, correntes, potências, factor de potência, frequência) de um dado analisador de energia, através da aplicação *ONLINE*. Por outro lado, permite registar os valores da energia activa e reactiva, de vários analisadores em simultâneo, separando automaticamente esta energia pelos vários períodos tarifários (cheia, vazio normal, super-vazio e ponta), através da aplicação *LOGGER*.

O sistema possibilita a análise dos parâmetros eléctricos que os analisadores permitem recolher de forma "*online*", criando com os valores gráficos e tabelas que facilitam monitorização dos consumos efectuados num complexo com inúmeros pontos críticos de consumo de energia. Disponibiliza funcionalidades como o envio dos dados recolhidos para contas de correio electrónico, impressão ou o simples arquivo dos dados recolhidos num ficheiro de forma a poderem ser posteriormente revistos e/ou analisados.

Através da aplicação "*LOGGER"* o sistema efectua o registo dos valores de consumos de energia activa e reactiva separando-os por analisador, reportando ainda toda a

informação recolhida a cada intervalo de 24h para um servidor onde são guardados todos os ficheiros diários. Estes dados podem por opção ser enviados também de forma periódica para endereços de correio electrónico.

Sendo a comunicação efectuada através do protocolo Modbus RTU, toda a execução é feita por Ethernet ou Internet o que favorece a implementação deste sistema em grandes complexos evitando a necessidade de considerar cablagem em toda a área. A cablagem tem custos de implementação extremamente elevados, sendo que por vezes implica que o sistema se torne inexequível.

A escolha do protocolo Modbus prende-se com o facto de ser um protocolo de livre acesso e, o mais utilizado na automação industrial. Aliado a isto está o facto de a maior parte das empresas terem aderido ao standard da Ethernet, o que possibilita a instalação de uma solução deste tipo sem ter que recorrer à passagem de novos cabos ou à reformulação da rede já existente no complexo.

Normalmente a implementação de sistemas de monitorização de consumos de energia utilizam conversores de rede RS-232 para rede RS-485, indicando a necessidade de um computador em cada rede implementada. O Energy Monitor permite a monitorização dos consumos de energia de várias redes necessitando apenas um computador para monitorizar todo o processo. Para tornar isso possível foram utilizados conversores RS-485/Ethernet, analisadores de energia e um PC normal onde são executadas as aplicações desenvolvidas em "*Visual Basic for Applications*" em execução no sistema operativo *Microsoft Windows*.

O sistema desenvolvido é flexível, apresenta uma interface amigável e uma grande capacidade de armazenamento de dados (usando o disco rígido do PC para o efeito). Tanto o hardware como o software contribuem para que se tenha obtido um sistema completo de baixo custo, uma vez que, para além do reduzido custo do equipamento, este sistema pode ser instalado utilizando redes Ethernet já existentes. A sua utilização em grandes complexos é assim facilitada pois eventuais limitações de distância são facilmente ultrapassadas.

Os objectivos apresentados no início deste exercício foram totalmente alcançados. O Energy Monitor foi testado, com sucesso, no campus da Faculdade de Ciências e Tecnologia onde continua a funcionar até hoje.

"Só quando a última árvore for derrubada, o último peixe for morto e o último rio for poluído é que o homem perceberá que não pode comer dinheiro."

Provérbio Indígena

5.2 Perspectivas de Trabalho Futuro

Em grandes complexos industriais, não é só a energia eléctrica que é consumida. Existem outros consumos, tais como o gás, a água ou energias renováveis. Assim sendo, este sistema poderá ser melhorado para incluir os registos relativamente a estas fontes de energia. Tornar-se-ia um sistema ainda mais completo e, onde o utilizador poderia controlar todos os custos.

Apesar da solução apresentada ser de custo reduzido, implica a utilização de um computador pois os analisadores de energia não têm memória interna. Uma possível sugestão para uma tese seria o desenvolvimento de um dispositivo com memória interna que centralizasse em si todos os dados dos diferentes analisadores espalhados pelo complexo. Desta forma, deixaria de ser necessária a presença de um computador. Quando o utilizador entendesse, acederia apenas a esse dispositivo para obter os dados que, seriam automaticamente tratados por uma aplicação.

Referências Bibliográficas

[1] Caruso, G. (2004). *Energy Information Administration Annual Outlook.* Estados Unidos da América: EIA.

[2] Protocol, K. (s.d.). United Nations Framework Convention on Climate Change.

[3] Nunes, P.Clemente. (2009). Uma melhor eficiência energética na indústria portuguesa.

[4] Caruso, G. (2008). Energy Information Administration Annual Outlook. Estados Unidos da América: EIA.

[5] Gaspar, C. (2009). Eficiência Energética na Indústria.

[6] CANopen. (2010). Manual da Comunicação.

[7] http://www.positronic.com.br/web/site/interbus.php (visto em 24/10/2009).

[8] Filho, C. S. (2000). Protocolos Orientados a Caracter.

[9] Curso de Redes Industriais: DeviceNet, Sense Eletrônica Ltda, 73 páginas ODVA, www.odva.com.br

[10] Halsall, F. Data Communications Computer Networks and Open Systems.

[11] Moraes, & Cirone. Redes de Computadores da Ethernet à Internet.

[12] http://www.ambienteonline.pt/noticias. (visto em 26/12/2009).

[13] http://www.energylens.com/oUPTuts. (visto em 26/12/2009).

[14] http://www.hughes-energy.com/MT/H-EMS.aspx. (visto em 26/12/2009).

[15] http://www.qenergia.pt/120/contagem-de-energia-wireless.htm. (visto em 26/12/2009)

[16] http://www.optimalcomms.net/OptimalSystem/Overview.aspx. (visto em 26/12/2009)

[17] Santos, T., Hagenfeldt, N., & Melo, R. (s.d.). Redes de Comunicação no Ambito da Automação Industrial.

[18] Tanenbaum, A. S. (2004). Computer Networks. Campus.

[19] Modbus-IDA. Modbus Application Protocol Specification.

[20] Lantronix. (2005). Modbus Protocol User Guide. Barranca Parkway.

[21] Makhija, J., & Subramanyan. (s.d.). Comparison of protocols used in remote monitoring.

[22] Modbus-IDA. (2004). www.Modbus.org. Obtido em 2009, de www.Modbus.org.

[23] Liao, G.-Y., Chen, Y.-J., Lu, W.-C., & Tsung-Chieh. (Outubro de 2008). Toward Authenticating the Master in the Modbus Protocol.

[24] Modicon. (1996). Modicon Modbus Protocol Reference Guide. North Andover: Modicon Inc.

[25] ApplicomIO. (s.d.). Modbus on Ethernet. applicom international

[26] Modbus-IDA. (s.d.). Modbus Application Protocol Specification V1.1b.

[27] Modbus RTU Unplugged (Na Introduction to Modbus RTU Networking)

[28] Modbus-IDA. (2004). www.Modbus.org. Obtido em 2009, de www.Modbus.org.

[29] ApplicomIO. (s.d.). Modbus on Ethernet. applicom international

[30] Lantronix. (2005). Modbus Protocol User Guide. Barranca Parkway.

[31] Elhadidy, & Shaahid. (2000). Parametric study of hybrid (wind+solar+diesel) power generating systems. Saudi Arabia.

[32] Jorge, H. (20 de Junho de 2008). Utilização Eficiente da Energia Eléctrica. Construção Sustentável e Eficiência Energética . Ourém, Portugal.

[33] Modbus-IDA. (2004). www.Modbus.org. Obtido em 2010, de www.Modbus.org.

[34] Algodue. (s.d.). Algodue Elettronica. Obtido em 2009, de http://www.algodue.com/index.php?alias=catalogue&act=products&id_parent=16

[35] Algodue. (2007). UPT210 Multi-Function Energy Meter. Technical Information

[36] LUMEL. (s.d.). RS-485/Ethernet Converter . RS-485/Ethernet Converter PD8 Type .

[37] Oliveira, R. S. RS-485 ou EIA-485.

[38] Goldie, J. (Outubro de 1996). Ten Ways to Bulletproof RS-485 Interfaces. National Semiconductor Corporation.

[39] Electrex. (s.d.). Technical Notes for RS-485 Network Installation. Itália.

[40] Algodue. (2009). UPT210 Multi-Function Energy Meter. Technical Information

[41] WinTECH. (2006). Industrial Automation Suite of Applications for the Windows

Anexo A – Varios tipos de protocolos de comunicação

Fonte: Synergetic Micro Systems

Fieldbus Name	Technology Developer	Year Introduced	Governing Standard	Openness
PROFIBUS DP/PA	Siemens	DP-1994, PA-1995	EN 50170 / DIN 19245 part 3(DP) /4 (PA), IEC 1158-2 (PA)	ASICs from Siemens and Profichip, Products from over 300 vendors
INTERBUS-S	Phoenix Contact, Interbus Club	1984	DIN 19258 EN 50.254	Products from over 400 manufacturers
DeviceNet	Allen-Bradley	March 1994	ISO 11898 &11519	17 chip vendors, 300+ product vendors, Open specification
ARCNET	Datapoint	1977	ANSI/ATA 878.1	Chips, boards, ANSI docs
AS-I	AS-I Consortium	Fall 1993	Submitted to IEC	AS-II.C. Market item
Foundation Fieldbus H1	Fieldbus Foundation	1995	ISA SP50/IEC 61158	Chips/software/products from multiple vendors
Foundation Fieldbus High Speed Ethernet (HSE)	Fieldbus Foundation	In development - lab test phase, Prelim spec available to members	IEEE 802.3u RFC for IP, TCP & UDP	Multitude of suppliers for Ethernet components, Extremely low cost
IEC/ISA SP50 Fieldbus	ISA & Fieldbus F.	1992 - 1996	IEC 1158/ANSI 850	Multiple chip vendors
Seriplex	APC, Inc.	1990	Seriplex spec	Chips available multiple interfaces
WorldFIP	WorldFIP	1988	IEC 1158-2	Multiple chip vendors
LonWorks	Echelon Corp.	March 1991		Public documentation on protocol
SDS	Honeywell	Jan., 1994	Honeywell Specification, Submitted to IEC, ISO11989	17 chip vendors, 100+ products
ControlNet	Allen-Bradley	1996	ControlNet International	Open Specification, 2 Chip Vendors
CANopen	CAN In Automation	1995	CiA	17 chip vendors, 300 product vendors, Open specification
Ethernet	DEC, Intel, Xerox	1976	IEEE 802.3, DIX v. 2.0	Multitudes of Chips and Products
Modbus Plus	Modicon			Proprietary, requires license/ASICs
Modbus RTU/ASCII	Modicon		EN 1434-3 (layer 7) IEC 870-5 (layer 2)	Open specification, no special hardware required

Remote I/O	Allen-Bradley	1980		Proprietary
Data Highway Plus (DH+)	Allen-Bradley			Proprietary

Fieldbus Name	Network Topology	Physical Media	Max. Devices (nodes)	Max. Distance
PROFIBUS DP/PA	Line, star & ring	Twisted-pair or fiber	127 nodes (124 slaves - 4 seg, 3 rptrs) + 3 masters	100m between segments @ 12Mbaud; 24 Km (fiber) (baudrate and media dependent)
INTERBUS-S	Segmented with "T" drops	Twisted-pair, fiber, and slip-ring	256 nodes	400 m/segment, 12.8 Km total
DeviceNet	Trunkline/dropline with branching	Twisted-pair for signal & power	64 nodes	500m (baudrate dependent) 6Km w/ repeaters
ARCNET	Star, bus, distributed star	Coax, Twisted-pair, Fiber	255 nodes	Coax 2000 feet; Twisted pair 400 feet; Fiber 6000 Feet
AS-I	Bus, ring, tree star, of al	Two wire cable	31 slaves	100 meters, 300 with repeater
Foundation Fieldbus H1	Star or bus	Twisted-pair, fiber	240/segment, 65,000 segments	1900m @ 31.25K wire
Foundation Fieldbus HSE	Star	Twisted-pair, fiber	IP addressing - essentially unlimited	100m @ 100Mbaud twisted-pair 2000m @ 100Mbaud fiber full duplex
IEC/ISA SP50 Fieldbus	Star or bus	Twisted-pair fiber, and radio	IS 3-7 non IS 128	1700m @ 31.25K 500M @ 5Mbps
Seriplex	Tree, loop, ring, multi-drop, star	4-wire shielded cable	500+ devices	500+ ft
WorldFIP	Bus	Twisted-pair, fiber	256 nodes	up to 40 Km
LonWorks	Bus, ring, loop, star	Twisted-pair, fiber, power line	32,000/domain	2000m @ 78 kbps
SDS	Trunkline/Dropline	Twisted-pair for signal & power	64 nodes, 126 addresses	500m (baudrate dependent)
ControlNet	Linear, Tree, Star, or Combination Thereof	Coax, fiber	99 nodes	1000m (coax) 2 nodes 250m with 48 nodes 3km fiber; 30km fiber w/ repeaters
CANopen	Trunkline/Dropline	Twisted Pair + optional Signal & Power	127 Nodes	25-1000m (baudrate dependent)
Industrial Ethernet	Bus, Star, Daisy-Chain	Thin Coax, Twisted Pair, Fiber; Thick Coax (rare)	1024 nodes, expandable to more via Routers	Thin: 185m 10 Base T (Twisted Pair): Max 100m long (90 metres horizontal cable, 5m drops, 1m patch) Max 4 hubs/repeaters between nodes 4Km distancs w/o routers Fiber: 100 Base FX 400m

				2.5 Km multi mode w/o Switches; 50 Km mono mode w/ Switches
Modbus Plus	Linear	Twisted Pair	32 nodes per segment, 64 max	500m per segment
Modbus RTU/ASCII	Line, star, tree Network w/ segments	Twisted Pair	250 nodes per segment	350m
Remote I/O	Linear Trunk	Twinaxial	32 nodes/segment	6 km
DH+	Linear Trunk	Twinaxial	64 nodes/segment	3 km

Fieldbus Name	Communication Methods	Transmission Properties	Data Transfer Size	Arbitration Method	Error Checking	Diagnostics
PROFIBUS DP/PA	Master/slave peer to peer	DP: 9.6, 19.2, 93.75, 187.5, 500 Kbps, 1.5, 3, 6, 12 Mbps PA: 31.25 kbps	0-244 bytes	Token passing	HD4 CRC	Station, module & channel diagnostics
INTERBUS-S	Master/slave with total frame transfer	500kBits/s, full duplex	1-64 Bytes data 246 Bytes Parameter 512 bytes h.s., unlimited block	None	16-bit CRC	Segment location of CRC error and cable break
DeviceNet	Master/slave, multi-master, peer to peer	500 kbps, 250 kbps, 125 kbps	8-byte variable message with fragmentation for larger packets	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	CRC check	Bus monitoring
ARCNET	Peer to peer	19.53K to 10M	0 to 507 bytes	Token passing	16-bit CRC	Built in Acknowledgements at Datalink layer
AS-I	Master/slave with cyclic polling	Data and power, EMI resistant	31 slaves with 4 in and 4 out	Master/slave with cyclic polling	Manchester Code, hamming-2	Slave fault, device fault
Foundation Fieldbus H1	Client/server publisher/ subscriber, Event notification	31.25 kbps	128 octets	Scheduler, multiple backup	16-bit CRC	Remote diagnostics, network monitors, parameter status
Foundation Fieldbus HSE	Client/Server, Publisher/Subscriber, Event Notification	100Mbps	Varies, Uses Standard TCP/IP	CSMA/CD	CRC	
IEC/ISA SP50	Client/server Publisher/ subscriber	31.25 kbps IS+1, 2.6, 5	64 octets high & 256 low	Scheduler, tokens, or	16-bit CRC	Configurable on network

Fieldbus		Mbps	priority	master		management
Seriplex	Master/slave peer to peer	200 Mbps	7680/transfer	Sonal multiplexing	End of frame & echo check	Cabling problems
WorldFIP	Peer to peer	31.25 kbps, 1 & 2.5 Mbps, 6 Mbps fiber	No limit, variables 128 bytes	Central arbitration	16-bit CRC, data "freshness" indicator	Device message time-out, redundant cabling
LonWorks	Master/slave peer to peer	1.25 Mbs full duplex	228 bytes	Carrier Sense, Multiple Access	16-bit CRC	Database of CRC errors and device errors
SDS	Master/slave, peer to peer, multi-cast, multi-master	1Mbps, 500 kbps, 250 kbps, 125 kbps	8-byte variable message	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	CRC check	Bus monitoring
ControlNet	Producer/Consumer, Device Object Model	5 Mbps	0-510 bytes variable	CTDMA Time Slice Multiple Access	Modified CCITT with 16-bit Polynomial	Duplicate Node ID, Device, Slave Faults
CANopen	Master/slave, peer to peer, multi-cast, multi-master	10K, 20K, 50K, 125K, 250K, 500K, 800K, 1Mbps	8-byte variable message	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	15 Bit CRC	Error Control & Emergency Messages
Industrial Ethernet	Peer to Peer	10, 100Mbps	46-1500 Bytes	CSMA/CD	CRC 32	
Modbus Plus	Peer to Peer	1Mbps	variable			
Modbus RTU/ASCII	Master/Slave	300 bps - 38.4Kbps	0-254 Bytes			
Remote I/O	Master/Slave	57.6 - 230 kbps	128 Bytes		CRC 16	none
DH+	Multi-Master, Peer<>Peer	57.6 kbps	180 Bytes			none

Fieldbus Name	Cycle Time: 256 Discrete 16 nodes with 16 I/Os	Cycle Time: 128 Analog 16 nodes with 8 I/Os	Block transfer of 128 bytes 1 node
PROFIBUS DP/PA	Configuration dependent typ <2ms	Configuration dependent typ <2ms	not available
INTERBUS-S	1.8 ms	7.4 ms	140 ms

DeviceNet	2.0 ms Master-slave polling	10 ms Master-slave polling	4.2 ms
ARCNET	Application Layer Dependent	Application Layer Dependent	Application Layer Dependent
AS-I	4.7 ms	not possible	not possible
Foundation Fieldbus H1	<100 ms typical	<600 ms typical	36 ms @ 31.25k
Foundation Fieldbus HSE	Not Applicable; Latency <5ms	Not Applicable; Latency <5ms	<1ms
IEC/ISA SP50	Configuration dependent	Configuration dependent	0.2 ms @ 5 Mbps 1.0 ms @ 1 Mbps
Seriplex	1.32 ms @ 200 kbps, m/s	10.4 ms	10.4 ms
WorldFIP	2 ms @ 1 Mbps	5 ms @ 1 Mbps	5 ms @ 1 Mbps
LonWorks	20 ms	5 ms @ 1 Mbps	5 ms @ 1 Mbps
SDS	<1 ms, event driven	5 ms polling @ 1 Mbps	2 ms @ 1 Mbps
ControlNet	<0.5 ms	<0.5 ms	<0.5 ms
CANopen	<1 ms	5 ms polling @ 1 Mbps	<2.5 ms
Industrial Ethernet	Application Layer Dependent	Application Layer Dependent	Application Layer Dependent
Modbus Plus			
Modbus RTU/ASCII			
Remote I/O	12msec @230, 40 msec @57.6 bus cycle time		
DH+			

Anexo B - Códigos de excepção

MODBUS Exception Codes		
Code	**Name**	**Meaning**
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, the PDU addresses the first register as 0, and the last one as 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 4, then this request will successfully operate (address-wise at least) on registers 96, 97, 98, 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 5, then this request will fail with Exception Code 0x02 "Illegal Data Address" since it attempts to operate on registers 96, 97, 98, 99 and 100, and there is no register with address 100.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
05	ACKNOWLEDGE	Specialized use in conjunction with programming commands. The server (or slave) has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client (or master). The client (or master) can next issue a Poll Program Complete message to determine if processing is completed.
06	SLAVE DEVICE BUSY	Specialized use in conjunction with programming commands. The server (or slave) is engaged in processing a long–duration program command. The client (or master) should retransmit the message later when the server (or slave) is free.
08	MEMORY PARITY ERROR	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check.
		The server (or slave) attempted to read record file, but detected a parity error in the memory. The client (or master) can retry the request, but service may be required on the server (or slave) device.
0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.